TURING 图灵程序设计丛书 Web开发系列

Peachpit Press



PHP for the Web Visual QuickStart Guide Fourth Edition

PHP基础教程

（第4版）

[美] Larry Ullman 著
贾菡 刘彦博 等译

- 经典PHP入门图书
- 循序渐进，示例丰富，图文并茂
- 体验轻松愉快的学习方式



人民邮电出版社
POSTS & TELECOM PRESS

图灵Web开发系列——PHP类书籍

PHP 6与MySQL 5基础教程

PHP与MySQL程序设计（第4版）

深入PHP：面向对象、模式与实践（第3版）

深入PHP与jQuery开发

PHP高级程序设计：模式、框架与测试

PHP实战

PHP与jQuery开发实例（即将出版）

高性能PHP应用开发（即将出版）

PHP编程实战（即将出版）



Larry Ullman 国际知名的技术作家，拥有20多年编程经验，精通多种语言和技术。他是DMC Insights公司的总裁和数字媒体技术总监，曾担任朗讯和Oracle等世界顶尖公司的顾问，并曾授课于加州大学伯克利分校。他撰写的《PHP 6与MySQL 5基础教程》（人民邮电出版社）等多部图书都广受世界读者欢迎，享有极高声誉。访问他的个人网站（www.LarryUllman.com）了解更多信息。

 图灵程序设计丛书

PHP基础教程
（第4版）
PHP for the Web

[美] Larry Ullman 著

贾 菡 刘彦博 等译

人民邮电出版社
北 京

图书在版编目（CIP）数据

PHP基础教程 : 第4版 / (美) 厄尔曼 (Ullman,L.) 著 ; 贾菡等译. -- 北京 : 人民邮电出版社, 2011.9
(图灵程序设计丛书)
书名原文: PHP for the Web : Visual QuickStart Guide
ISBN 978-7-115-26093-2

Ⅰ. ①P… Ⅱ. ①厄… ②贾… Ⅲ. ①PHP语言—程序设计—教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2011)第152449号

内 容 提 要

本书以通俗易懂的语言向初学者介绍了 PHP 语言的基本概念、使用方法和注意事项。全书通过丰富的示例，引领读者逐步掌握这门流行的 Web 开发语言，使读者能够上手编写适用于常用场景的 PHP 脚本。

本书适合有基本 HTML 经验的读者阅读。

图灵程序设计丛书
PHP基础教程（第4版）

◆ 著 [美] Larry Ullman
译 贾 菡 刘彦博 等
责任编辑 傅志红
执行编辑 刘美英

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京 印刷

◆ 开本：800×1000 1/16
印张：26.5
字数：659千字 2011年9月第1版
印数：1－3 000册 2011年9月北京第1次印刷
著作权合同登记号 图字：01-2009-6903号
ISBN 978-7-115-26093-2

定价：65.00元

读者服务热线：(010)51095186转604 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

致　谢

非常感谢Peachipt出版社的每一个人，感谢他们的协助和辛苦的工作。特别感谢以下人员。

非常优秀的编辑Rebecca Gulick，感谢她做的每一件事。

感谢Liz Welch对本书所有细节的关注。

感谢Jay Blanchard为本书做技术审校，他有一种很神奇的能力，总能预知我接下来要表达的东西。

感谢Bob Campbell为本书做校对，他的目光非常敏锐。

感谢Deb Roberti和Myrna Vladic把一大堆零散的资料整理成册。感谢Valerie Haynes-Perry为本书制作索引。

衷心感谢阅读本书其他版本和我的其他书籍的读者，感谢你们长期以来对我工作的反馈和支持。

感谢Rasmus Lerdorf（PHP之父）、PHP.net和Zend.com的朋友、经常访问新闻组和邮件列表的朋友，以及为这项伟大技术提供开发、改进和支持的强大的PHP开源社区。

感谢Karnesha帮我照顾孩子们，让我有足够的时间完成撰写工作，不过我真是不太喜欢照顾孩子。

感谢我的孩子Zoe和Sam，他们一直都很讨人喜爱。

感谢Jessica所做的所有工作，她把这些混乱的工作做得井井有条，并且把一切都考虑得仔细周到。

引　　言

2000年我在编写本书第一版时，PHP还只是一个几乎不为人知的开源项目。它被熟知内幕的技术人员所钟爱，但是还没有像今天这样成为Web开发方面公认流行的选择。在我自学PHP的时候，关于这种语言的文档少之又少，这正是我编写此书的初衷。

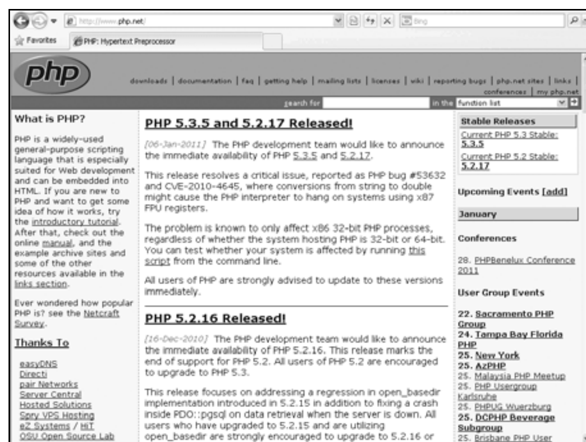
现在形势不一样了。因特网经历大起大落之后，进入了稳定的发展期。而且，PHP现在已经成为程序员首选的动态Web设计工具，并且开始将领域扩张至Web开发之外。但是，尽管PHP已经广为流行，相关的文档、示例代码和样例也越来越丰富，出版一本介绍PHP的好书还是很有必要的。PHP已经发布了5个主要版本，对于学习PHP的读者来说，本书这样简洁实用的教程正是所需的最佳指南。

本书不仅帮助读者深入理解基本原理，同时还将引导读者获取进阶信息。尽管本书不是编程参考大全，但是通过详尽阐述和真实样例，为读者提供了使用PHP构建动态Web站点和Web应用程序的必备知识。

PHP是什么

PHP起初是Personal Home Page的缩写，意为个人主页。它最早是由Rasmus Lerdorf在1994年创建的，用来跟踪他本人在线简历的访问者。随着PHP的实用性和功能的扩展（同时它被应用在越来越专业的场景中），它代表的意思变化为PHP：Hypertext Preprocessor（PHP，超文本预处理程序）。[这个定义的主要意思是PHP在数据变为HTML（HyperText Markup Language）之前先处理数据。]

通过PHP的官方网站www.php.net（参见图i-1）可以了解到，PHP是一种HTML内嵌式脚本语言。下面详细解释这个定义的含义。



图i-1　该图为编写本书时PHP官方网站的截图，网址是www.php.net。这是寻找关于PHP问题的答案和满足对它好奇心的首选地址

PHP定义中“HTML内嵌式”的意思是它可以混杂在HTML代码中。HTML是一种用来生成所有Web页面的代码。因此，使用PHP编写代码只比使用HTML稍微复杂一点点。

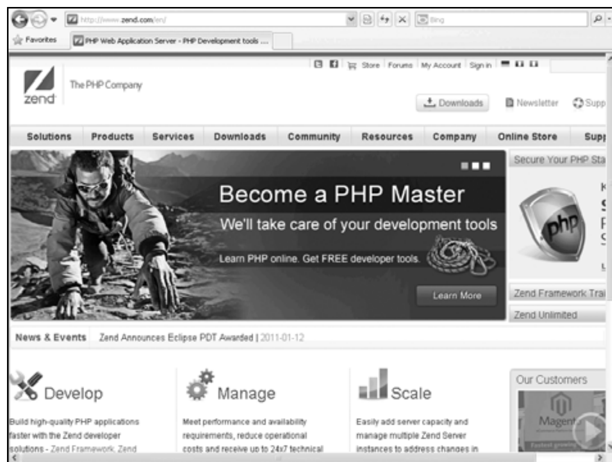
此外，相对于*编译语言*来说，PHP是一种*脚本语言*。也就是说PHP被设计成为仅*当事件发生*之后才会开始工作。例如在用户提交表单后，或者前往某个URL（Uniform Resource Locator，统一资源定位符，即Web地址）时，PHP才会开始工作。另一种流行的脚本语言就是JavaScript，它被普遍应用于处理Web浏览器中发生的事件。这两种语言都被描述成解释型语言，因为它们的代码都要通过一个可执行文件（比如PHP解释器或浏览器的JavaScript引擎）来解释执行。相反，像C和C++那样的编译语言可以用来编写独立的应用程序，编译后可直接运行。

PHP还是一种*服务器端技术*。这是指PHP所做的一切均在服务器端（而不是在*客户端*——用户浏览网站的计算机中）发生。*服务器*是一台计算机，用来提供用户使用浏览器（例如，Firefox、IE或者Safari）访问某个Web地址时的页面。后文将介绍这个过程的细节（参看“PHP是如何工作的”一节）。

最后，PHP是一种*跨平台*的技术，也就是说它能够用在运行Unix、Windows、Macintosh和其他操作系统的机器上。当然我们讨论的是*服务器*运行的操作系统，而不是客户端运行的操作系统。PHP不仅能够运行在几乎所有的操作系统上，而且与其他大多数的编程语言不同，它能够在不同的操作平台上进行切换，而不做或者仅仅做很少的修改。

在撰写本书时，PHP的版本为5.3.5和5.2.17（5.3和5.2两个版本之间差别不是很大，因此在一段时间内5.2还将获得支持）。尽管本书的代码环境使用稳定的PHP 5.3，但是所有的代码都能够向后兼容。如果不能兼容4.x版，至少能够支持PHP的5.x版。本书偶尔会使用最近版本的PHP，可能会与旧版本有些许不同，此时会在注解或者提示中说明如何相应地调整代码。

请访问PHP.net和www.zend.com以获得更多信息，zend.com体现了PHP的核心思想（参见图i-2）。



图i-2　这是Zend的主页，聚集了编写PHP的核心开发人员。该网站包含了大量有用的软件、代码库和编写得很好的教程

PHP的局限性

学习PHP的新手们经常感到迷惑不解的问题是：PHP不能用来做什么。虽然可以用这门语言来完成纷繁的任务，但是它最主要的限制就是不能在Web站点中实现客户端的功能。

使用JavaScript这样的客户端技术，可以创建一个新的浏览器窗口、添加鼠标悬停响应、弹出警告窗口、重设浏览器窗口的大小、获取用户机器的屏幕尺寸，并且动态地生成和修改表单。这些用PHP都无法做到（因为PHP是服务器端脚本语言，而上述问题都需要在客户端实现）。但是，可以用PHP生成JavaScript，就像可以用PHP生成HTML那样。

在开发自己的PHP项目时，请记住只能使用PHP向Web浏览器发送信息（HTML等）。在向服务器发送另一个请求之前（比如提交表单或者单击某个链接之前），不能在Web浏览器中做任何事情。

为什么要使用PHP

简而言之，与其他同类语言相比，PHP能够表现得更好、更快，并且更简单易学。所有的Web站点都必须以HTML开始，因此可以使用许多静态HTML页面来创建一个完整的站点。但是基础的HTML在灵活性和提供响应方面都有局限。访问者进入HTML页面时看到的是简单的、没有定制或者动态行为的页面。使用PHP则可以进行同数据库和文件的交互、处理邮件等操作，做很多HTML不能做的事情。

很久以前，Web站点设计者就认识到不能单独用HTML创造出迷人、持久的Web站点。为了结束这种状态，诸如PHP的服务器端技术应运而生。这些技术能够让Web页面设计者创建Web应用程序，这些Web应用程序能够动态地生成，并且将程序员所渴望的元素都纳入考虑的范畴。这些高级站点通常都是数据库驱动的，能够比静态HTML页面更快地升级和维护。

当选择服务器端技术时，PHP最主要的对手是CGI脚本（Common Gateway Interface，通用网关接口，并不一定用Perl编写）、ASP.NET、Adobe的ColdFusion、JSP（JavaServer Pages）和Ruby on Rails。因为JavaScript是一种客户端技术，并且不能同PHP或其他这类技术一样用来创建HTML页面，所以JavaScript并不是PHP的真正替代品（反之亦然）。

现在问题是，为什么Web设计者要用PHP而不用CGI、ASP.NET、JSP等技术来创建动态的Web站点呢？

- **PHP更易于学习和使用**。在阅读本书之后，没有经过正式编程培训的人都能够轻松地编写PHP脚本。相比较而言，ASP.NET要求了解VBScript、C#或者其他语言；CGI要求有Perl（或者C）编程基础。它们都是更加复杂且难以学习的语言。
- **PHP专门用来编写动态Web页面**。Perl（以及VBScript和Java）则不是，这个事实暗示了出于自身特定的意图，PHP能够在处理特定任务时比它的那些竞争者更加迅速、更加简便。但是本书需要点明的是，虽然在处理某些任务时更加优秀（因为它就是为了解决这些问题而创造出来的），PHP并不是比Java或者Perl更好的编程语言——后两者能够做许多PHP

不能做的事情。

- **PHP不仅免费而且跨平台**。因此，可以在任何计算机上使用它而不产生费用。此外，PHP的开源本质上意味着是PHP的用户推动了它的发展，而不是由某个企业实体推动的。
- **PHP是可以用来开发动态Web站点的最受欢迎的工具**。在编写本书时，有超过75%的站点是由PHP编写的（参见图i-2），在所有的编程语言中，受欢迎程度上，PHP排在第4位（参见图i-3）。很多大型Web网站（如Yahoo!、Wikipedia和Facebook等）和内容管理工具（如WordPress、Drupal、Moodle和Joomla）都使用PHP。掌握这项技术，你不仅发展了一项实用的业余爱好，同时也掌握了一门可使自己获利的技能。

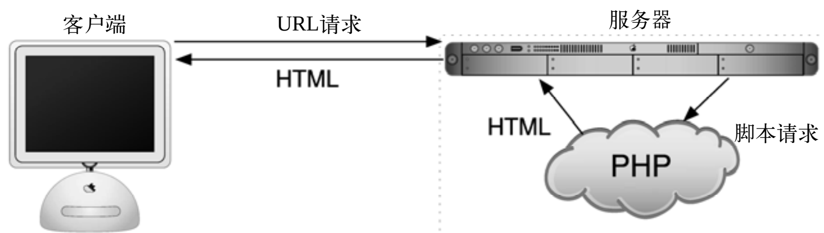
Position Jan 2011	Position Jan 2010	Delta in Position	Programming Language	Ratings Jan 2011	Delta Jan 2010	Status
1	1	=	Java	17.773%	+0.29%	A
2	2	=	C	15.822%	-0.39%	A
3	4	↑	C++	8.783%	-0.93%	A
4	3	↓	PHP	7.835%	-2.24%	A
5	7	↑↑	Python	6.265%	+1.81%	A
6	6	=	C#	6.226%	+0.46%	A
7	5	↓↓	(Visual) Basic	5.867%	-1.49%	A
8	12	↑↑↑↑	Objective-C	3.011%	+1.63%	A
9	8	↓	Perl	2.857%	-0.71%	A
10	10	=	Ruby	1.784%	-0.69%	A
11	9	↓↓	JavaScript	1.589%	-1.12%	A

图i-3 Tiobe Index（http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html）对流行的编程语言的综合排名。

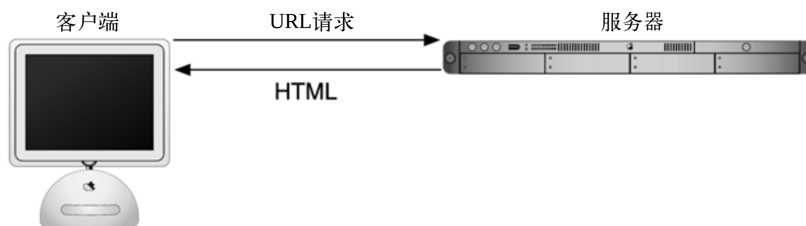
PHP是如何工作的

PHP是一种服务器端语言，这意味着用PHP编写的代码将在为Web浏览器提供Web页面的主机上运行。当访问一个Web站点（例如，www.LarryUllman.com）时，所涉及的Internet服务提供者（ISP）将把请求定向到保存着www.LarryUllman.com信息的服务器上。服务器读取PHP代码并执行脚本指令。在这个示例中，PHP代码告知服务器以HTML的形式向浏览器发送适当的Web页面（参见图i-4）。简而言之，PHP按照所选择的参数创建了一个HTML页面。

与HTML生成的网站有所不同，当请求发出时，服务器仅仅向Web浏览器发送HTML数据——没有服务器端解释发生（参见图i-5）。换句话说，在最终用户的浏览器上查看home.html和home.php并不一定有明显的区别，但是如何生成这两个页面却有很大不同。主要的不同之处在于，使用PHP可以让服务器动态地生成HTML代码。例如，今天是星期一而不是星期二，或者如果用户已经访问过该页，这样的不同信息能够呈现出来。动态Web页面的创建，将不那么吸引人的静态网站同更有趣因而访问量更大、更具有交互性的网站区分开来。



图i-4　这张图示范了在客户端和服务器端之间进行的处理工作（虽然是在极其简单的条件下），PHP模块（添加进服务器的一个应用程序，用来增加它的功能）用来向浏览器发回HTML。所有的服务器端技术都在服务器上使用一个第三方模块，用来处理发送回客户端的数据



图i-5　将图i-5中服务器处理HTML的方式与图i-4进行比较。这跟通过浏览器查看本地的HTML页面没有区别——本地页面不需要服务器处理，但是动态生成的页面需要通过服务器访问，因为动态页面要经过服务器处理

使用PHP和直接使用HTML之间重要的不同之处在于，PHP在服务器端处理完所有的事情之后向浏览器发送适当的信息。本书将介绍如何使用PHP向浏览器发送正确的数据。

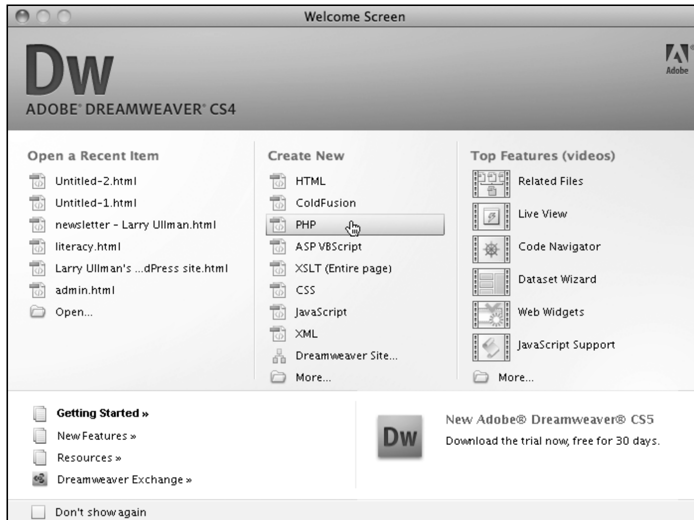
准备工作

使用PHP最重要的条件是能够访问启用了PHP的服务器，这是因为它是一种服务器端脚本语言。由于PHP相当普及，因此ISP或者Web主机提供商极有可能已经在他们的服务器上安装了PHP模块。使用时需要同他们联系以确认支持哪些技术。

另外一个选择是，在自己的计算机上安装PHP和Web服务器应用程序（如Apache）。Windows、Mac OS X或者Linux的用户都能够很容易地免费安装和使用PHP。附录A提供了安装PHP的指导。如果希望使用自己的服务器，并且自己安装PHP，那么可以在PHP的Web站点（www.php.net）上找到免费下载的安装包。如果采用后一种方式（这也是我们推荐的方式），那么你的计算机将可以同时作为客户端和服务器端使用。

第二个条件是，在计算机上必须拥有一个文本编辑器。Crimson Editor、SciTE、TextWrangler以及类似的免费应用程序都能够很好地满足需要，同时BBEdit、EditPad、TextMate和其他的商业应用程序可以提供用户喜欢的更丰富的特性。如果习惯使用如Adobe Dreamweaver（参见图i-6）

或Aptana Studio的图形化界面（也被称作WYSIWYG——所见即所得），可以考虑查阅这些应用程序的手册以了解如何利用它们进行编程。



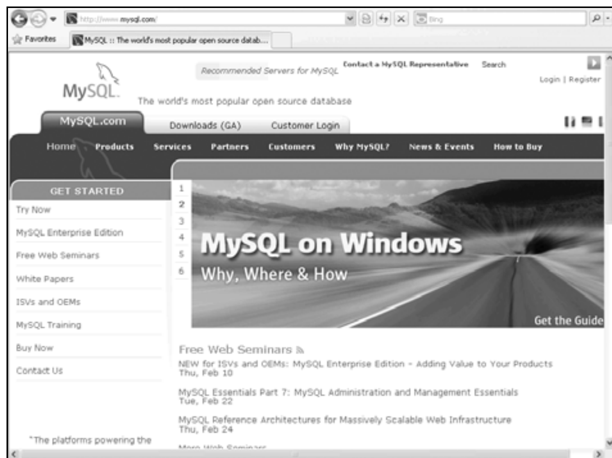
图i-6　广受欢迎的Dreamweaver IDE提供包括PHP在内的服务器端技术的开发环境

第三，PHP是一种把在文本编辑器中编写的脚本放到服务器上的方法。如果在自己的计算机上安装了PHP，那么可以将它们保存在适当的目录中。但是，如果使用ISP或者Web主机提供商提供的远程服务器，那么需要一个FTP（File Transfer Protocol，文件传输协议）应用程序把脚本上传到服务器。在第1章中，本书将使用免费的FileZilla（www.filezilla-project.org，参见图i-7）作为示例。



图i-7　FileZilla应用程序可以运行在许多操作系统上，用于向远程服务器上传PHP脚本和其他的文件

最后，如果读者想试验第12章中的示例，那么使用MySQL（www.mysql.com，参见图i-8）或者其他数据库应用程序。MySQL也是免费的应用程序，可以安装在自己的计算机上。



图i-8　（在编写此书时）MySQL的Web站点

本书假定读者仅具备HTML的基础知识。当然，如果比较熟悉手工编写HTML代码，而不是使用诸如Dreamweaver这类创建Web网站的辅助应用程序，那么学习PHP将变得更加容易。每个程序员都会需要查找HTML参考资料，不论之前已经了解多少相关知识，我都建议你将一本好的HTML参考书放在手边。这类HTML编程入门书籍有Elizabeth Castro编著的《HTML XHTML CSS基础教程（第6版）》（人民邮电出版社2007年出版）。

本书并不要求读者具备编程的经验。但是，如果你有编程经验将会加速学习的进度，因为在学习的过程中你会很快发现有很多地方都同以往的经验相通。例如，Perl和PHP或者JavaScript和PHP都有类似之处。

关于本书

本书旨在向读者介绍PHP编程的基础知识，同时也对他们将来可能用到的更多高级特性给出提示，而没有过分探究细节。本书中的内容将延续下面的惯例。

逐步介绍部分指明了需要你手工编写的代码，以及该代码在脚本中的位置。这些需要输入的代码以下面字体给出。例如：

```
<?php print "Hello, World! "; ?>
```

本版的新内容

第4版是对第3版的进一步完善。这一版变化最大的是第13章，删除了旧版正则表达式的全部内容。本书前几版讨论的正则表达式的类型，已在新版本的PHP中废弃，也就是说PHP对正则表达式的支持会越来越少。其他处理正则表达式的方法非常复杂，不适合初学者学习。在我

撰写的《PHP 6与MySQL 5基础教程》[1]一书中，详细说明了正则表达式的使用方法。

新版的第13章是全新的一章。在这一章中我将带你一步步创建一个功能齐全的网站，这将会用到本书介绍的所有知识（还有一些小技巧）。我希望通过实战，这个全新的章节能帮你快速掌握新学的知识。

除此之外，这一版各章都增加了“回顾和实践”小节，篇幅一到两页。我会问些问题，帮你加深对每章重要知识点的理解；同时也列出了将要学习的相关内容、附加信息或一些练习。问题的答案和提示可在本书的论坛中找到，网址www.LarryUllman.com/forum/。

最后，为力求完美，我修订了一些示例。

PHP代码还会以完整的脚本给出，并且带有行号作为参考（参看脚本i-1）。读者在编写代码时不能插入这些数字，因为这样将导致代码不能执行。本书建议使用能自动显示行号的文本编辑器——在进行调试工作的时候，这些数字可以提供帮助。有时候，你会发现脚本中某些特殊的行被加粗了，这是为了引起你对新的或者其他相关信息的注意。

脚本i-1　PHP示例脚本，标记有行号，并且对特殊代码段用粗体强调

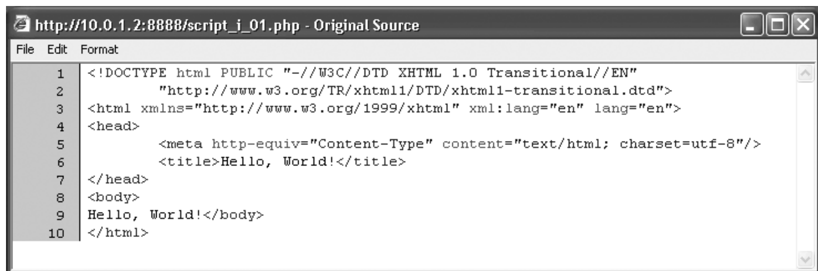
```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Hello, World!</title>
7   </head>
8   <body>
9   <?php print "Hello, World!"; ?>
10  </body>
11  </html>
```

根据PHP的工作原理，每段脚本都可以从3个角度来观察：PHP代码（如脚本i-1）、发送给浏览器的代码（主要是HTML）以及浏览器向最终用户的呈现效果。在适当情况下，本书将通过部分或完整的浏览器窗口的屏幕截图，呈现脚本示例运行的最终结果（参见图i-9）。有时候也会给出浏览器显示接收到的HTML源代码的截图（参见图i-10）。通常可以在适当的Web浏览器菜单中，选择View Source或者View Page Source来查看源代码。总之，图i-10显示了浏览器接收到的HTML，而图i-9演示了浏览器解释相应HTML的结果。使用PHP，可以创建发送到浏览器的HTML。



图i-9　这是一个示例视图，可以看到浏览器窗口。对于本书中的示例，无论你使用什么浏览器或操作系统，结果都不会有什么不同

[1] 本书已由人民邮电出版社于2008年10月出版。——编者注



图i-10　查看Web浏览器获取的源代码，可以看到由服务器发送的PHP创建的HTML

由于版面有限，因此逐步介绍中的某些PHP代码行在印刷时会出现折行，这种情况也许在读者的编辑器中不会出现。在出现这样的情况时，本书会用一个小灰色箭头表示折行。例如：

```
print "This is going to be a longer line
→of code.";
```

对此，读者在脚本中的代码应该还是写在一行中，否则在执行时将会遇到错误（编有行号的脚本中不使用灰色箭头）。

当示范新特性和新技术时，本书将尽力解释清楚“为什么这样”以及“如何做到”。通过阅读理解和使用函数，应该可以确切地掌握它。当然，有一些疑问是不可避免的，本书为此提供了一些参考资源以便查找到问题的答案（参看附录B）。如果对某些特定的函数或者示例有疑问，最好查看在线的PHP手册，或者参考本书提供的Web站点（以及其中的用户支持论坛）。

哪本书更适合

这是我为PHP写的第一本书的第4版。同第1版一样，本书的编写考虑到了初学者和没有编程基础的读者。如果只有很少甚至没有编程经验，或者偏爱舒缓的学习进度、喜欢学习的时候慢慢咀嚼，那么这本书最适合不过了。请别误会：本书其实涵盖了开发动态Web网站必备的所有知识（还使用了真实的示例），但是它并不包含深奥的理论或者高级应用。

相反，如果能够迅速接受新技术，或者已经有了一定程度的Web站点开发经验，你可能会觉得本书太基础了。在这种情况下，可以考虑阅读我的《PHP 6与MySQL5基础教程》。该书详细讨论了SQL和MySQL，并且提供了不少更加复杂的示例，而且其学习步调非常轻快。

关联网站

本书关联的Web站点非常有用，其网址是www.LarryUllman.com。在这里，可以下载书中涉及的每段代码[1]。（但是，本书强烈建议读者自己输入这些代码，这样对熟悉PHP的结构和语法都非常有帮助。）

这个网站还包括一个更加详细的参考部分，这个部分提供了大量Web页面的链接，通过这些

① 本书代码也可以从图灵网站www.turingbook.com本书网页免费注册下载。——编者注

页面可以进一步学习PHP。此外，站点的勘误部分还列出了本书中存在的错误。

本书的支持论坛可能会给读者带来最多帮助，论坛的地址是www.LarryUllman.com/forum。这个论坛可以帮你：

- 找到问题的答案；
- 得到如何实现你的想法的建议；
- 获得来自其他读者的调试帮助；
- 了解这项技术的变化给本书中的示例带来怎样的影响；
- 学习其他人如何使用PHP；
- 获得复习中的问题答案；
- 获得比直接给我发送邮件更快的回复。

问题、意见和建议

如果有与PHP相关的问题，可以通过新闻组、邮件列表，以及PHP有关的Web网站中的问答区来解决。附录B更详细地讨论了这方面的内容。浏览这些参考资料或者在Internet上搜索通常是最快获得问题解答的方式。

也可以将问题、意见和建议直接发送给我。使用本书的关联论坛，将获得最快的答复（我通常会首先在这里回答问题）。如果更希望用邮件的方式，我的联系信息在Web网站上也可以找到。我将尽力回复收到的每封邮件，但也许要等上几周（如果在论坛上提问，可能只要等几天就可以收到回复）。

如果希望获得更多的提示和启蒙式的阅读，请在www.catb.org/~esr/faqs/smart-questions.html上参看Eric Steven Raymond所写的文章的“How to Ask Questions the Smart Way”。花10分钟来阅读该文章，可以更快速地得到问题的答案。包括我在内的其他将回答问题的人们会不胜感激！

提问的智慧

不论是向本书的支持论坛发布消息、给我发送邮件，还是在新闻组里问问题，知道如何最有效地提问将大大提升得到回复的质量，以及获得答案的速度。为了在最短时间内获得到最佳回答，请遵照以下的步骤：

(1) 在Internet上搜索，阅读相关手册，浏览所有可能有用的文档；

(2) 在最合适的论坛（新闻组、邮件列表等）上提问；

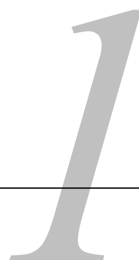
(3) 使用清晰简明的标题；

(4) 详细描述你的问题，提供相关的代码，告知什么地方出错，以及使用的PHP版本和操作系统。

目　　录

第 1 章

PHP概述



本章内容

- ❑ HTML语法基础
- ❑ PHP语法基础
- ❑ 使用FTP
- ❑ 测试脚本
- ❑ 向浏览器发送文本
- ❑ 使用PHP手册
- ❑ 向浏览器发送HTML
- ❑ 为脚本添加注释
- ❑ 调试的基本步骤
- ❑ 回顾和实践

学习任何新的编程语言时，首先需要理解的是它的基础语法，这也正是本章介绍的内容。本书将重点讨论PHP的基础知识，但是同时也会涉及一些备受推崇的编程技术，从长远来看，这些技术能够改进我们的工作。

如果读者之前没有任何编程经验，那么应该仔细阅读本章，它将为你指明今后学习的正确方向。如果已经具备了一定的编程经验，那么可以快速浏览这一章，了解本书其余的内容。在结束本章时，读者就将能够成功地编写并执行第一个PHP脚本，走上开发动态Web应用程序之路。

1.1 HTML 语法基础

所有的Web页面都由HTML（HyperText Markup Language，超文本标记语言）构成。每种Web浏览器（无论是微软的IE、苹果的Safari、Mozilla的Firefox还是Google的Chrome）都将HTML代码：

```
<h1>Hello, World!</h1>
I just wanted to say <em>Hello</em>.
```

转变成为有样式的Web页面展示给用户（参见图1-1）。



图1-1　Web浏览器呈现HTML代码的方式

在撰写本书时，HTML的最新版本是4.01。下一个主要版本是HTML 5，目前正在积极地研发和讨论中，还没有最后定案。本书使用XHTML（eXtensible HTML），它与HTML有少许差别。实际上，XHTML同HTML非常相似，它们之间的不同之处有以下几点。

- 所有的标签都使用小写字母。
- 嵌套标签必须有恰当的格式。

  这条规则并不像它听上去那么复杂，它的意思是，不能这样写代码：`<div><p>text </div></p>`，而应该使用这种格式：`<div><p>text</p></div>`。
- 所有的标签属性值必须使用引号引起来。

  在HTML中可以这样写：`<table border=2>`，但是在XHTML中必须这样写：`<table border="2">`。
- 所有的标签必须关闭。

  这条规则最容易让大多数人感到迷惑。许多HTML标签同时拥有开启和关闭标签，如`<div class="someclass">text</div>`。但是有一些HTML标签并不强制使用关闭标签，包括`<hr>`、`<br>`、`<img>`和`<input>`。为了编写有效的XHTML标签，必须在末尾处添加一个空格和斜线以关闭它们，就像这样：

```
<hr />
<br/>
<img src="image.png" />
<input type="text" name="age" />
```

CSS基础

HTML和XHTML元素定义页面的内容，但要格式化这些内容的外观和行为就要依赖CSS（Cascading Style Sheet，层叠样式表）。由于主题是围绕HTML和XHTML的，因此本书不会涉及太多CSS细节，但是本书有时会在代码中用到CSS，你最好了解一些CSS的基础语法。

有两种方法可以在Web页面中加入CSS。第一种方法（推荐）是使用HTML `style` 标签：

```
<style type="text/css">
CSS规则
</style>
```

在开始标签和结束标签之间定义CSS规则。第二种方法是使用`link`标签引入在外部文件中定义的CSS规则：

```
<link href="styles.css" rel="stylesheet" type="text/css" />
```

CSS规则可应用于一般页面元素、CSS类和特定元素：

```
img { border: 0px; }
.error { color: red; }
#about { background-color: #ccc; }
```

以上代码中，第一条规则应用于所有图片标签。第二条规则应用于所有class属性为error的元素：

```
<p class="error">Error!</p>
```

第三条规则应用于id值为about的所有特定元素：

```
<p id="about">About...</p>
```

（不是所有的元素都要有id属性，而且任意两个元素的id值不能相同。）

多数情况下，本书只使用CSS实现一些非常简单的样式，例如改变颜色以及一些元素或文本的背景色。

虽然使用单独的CSS段或文件是最好的方法，但为了简化工作，本书有时会直接在代码中加入CSS：

```
<p style="color: red;">Error!</p>
```

你可以在网上搜索或查阅专门书籍，了解更多CSS相关知识。

在深入讲解PHP语法之前，让我们创建一个简单但是有效的XHTML文档，它将作为本书中几乎所有示例的模板。

⇛创建XHTML页面

(1) 打开文本编辑器或者IDE。

可以使用几乎任何应用程序来创建HTML、XHTML和PHP页面。通常的选择包括有：Adobe的Dreamweaver（www.adobe.com）（运行在Windows和Mac OS X上）、EditPlus（www.editplus.com）和Crimson Editor（www.crimsoneditor.com）（运行在Windows上），以及运行在Mac上的Bare Bones的BBEdit（www.barebones.com）或者MacroMates的TextMate（www.macromates.com）。

(2) 选择文件>新建来创建一个新的空白文档。

一些文本编辑器可以从创建一个某种类型的新文档开始，如一个新的XHTML文件（参见图1-2）。如果所使用的应用程序有这样的选项，敬请使用它。

(3) 从XHTML头行开始（参看脚本1-1）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

脚本1-1 示例文档显示了基本的XHTML代码

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

```
6      <title>Welcome to this Page!</title>
7    </head>
8    <body>
9    <h1>This is a basic XHTML page!</h1>
10   <br/>
11   <p>Even with <span style="font-size: 150%;">some</span> decoration, it's still
     not very exciting.</p>
12   </body>
13   </html>
```

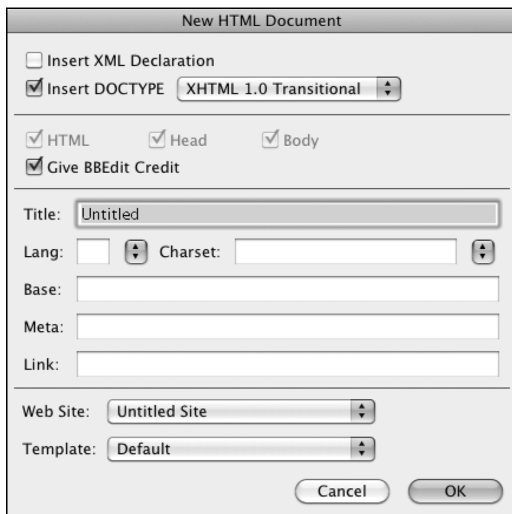


图1-2　BBEdit和其他大多数Web开发应用程序一样，都可以用来创建基本的XHTML文档

一个有效的XHTML文档都以这几行开头，它们用来告诉Web浏览器文档的类型。在本书这个模板中，将创建XHTML 1.0*过渡*页，这就是说将遵循XHTML 1.0的标准。过渡部分的意思是这里将允许使用*不推荐*的（不再被建议）标签（同严格模式相反，在严格模式中这是不允许的）。

(4) 创建页面的head部分：

```
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Welcome to this Page!  </title>
</head>
```

XHTML页面的head部分包含content-type元标签（有效的XHTML所必需的）和title标签。框注“理解编码”讨论了标签中charset部分的意思。JavaScript和CSS引用也放在这里。

(5) 创建body部分：

```
<body>
<h1>This is a basic XHTML page!</h1>
<br/>
<p>Even with <span style="font-size: 150%;">some</span> decoration,
→it's still not very exciting.</p>
</body>
```

在Web浏览器中所看到的页面的内容包含在开始和关闭的body标签之间。遵循XHTML的规则，换行标签（
）包含一个空格和紧随其后的斜线，用来关闭标签。所有的其他标签都同它们在标准HTML中对应的标签一样，只不过要使用小写字母编写。其中的CSS代码用来增大some的字号。

(6) 输入</html>以完成这个HTML页。

(7) 选择文件>另存为。当对话框出现时，如果提供了选项，请选择TextOnly（或者ASCII）格式。XHTML和PHP文档都是纯文本文件（这同Microsoft Word文档专有的格式不同）。在保存文件时，还可能需要指明编码（请参看框注“理解编码”）。

(8) 导航至希望保存脚本的位置。

尽管在本书中会将每个脚本保存在各自专有的文件夹里，但是读者可以将脚本放置在计算机上的任何地方，当然为每章建立子文件夹是非常必要的。

(9) 将文件保存为welcome.html。

虽然这里是使用XHTML进行编码，但是页面将仍然使用标准的.html或者.htm扩展名。

(10) 在Web浏览器上查看该页以进行测试（参见图1-3）。

同PHP脚本不同（读者很快就能发现），可以通过在Web浏览器中直接打开XHTML和HTML页面测试它们。



图1-3　XHTML文档被Web浏览器解释显示

理解编码

编码是一个复杂的主题，但是当前最需要理解的是：在某文件中所使用的编码将指明可以显示什么字符（因此也同时指明可以使用哪种语言）。在选择一种编码之前，必须首先确定所使用的文本编辑器或者IDE能够将文档保存为这种编码格式。一些应用程序可以让你在偏好设置或者选项中设置编码，另外一些应用程序只有当保存文件时才可设置编码。

向Web浏览器指明选用哪种编码，可以使用相应的元标签：

```
<meta http-equiv="content-type" content="text/html;  charset=utf-8" />
```

charset=utf-8部分表明使用UTF-8（8-bit Unicode Transformation Format，8位Unicode转换格式）编码。Unicode是一种能够显示任何字母表中每个符号的可靠方法。Unicode第6版（在编写此书时的最新版）支持的字符超过99 000个！最常使用的Unicode编码是UTF-8。

如果你需要创建一个多语言的Web页面，UTF-8将是很好的选择。本书中的示例会使用它，虽然并不是必须这样做。但是，无论你使用哪一种编码，都要确保XHTML页面中指定的编码与文本编辑器或IDE实际使用的编码相匹配，否则你可能会在使用Web浏览器浏览页面时看到一些奇怪的字符。

✓提示

- 请访问www.LarryUllman.com/forum/，查找适用于HTML和PHP的编辑器或者IDE。
- 本书使用XHTML，但是这并不意味着你也必须这样做。如果感觉HTML更加顺手，请坚持使用了解的语言。这样不会影响PHP脚本运行。
- 参阅Elizabeth Castro的优秀著作《HTML XHTML CSS基础教程（第6版）》以了解更多关于XHTML、XML和HTML的信息。
- 本书将混用HTML和XHTML。事实上，大多数时候你将只看到HTML，但是请注意，实际上它指的是XHTML。

1.2　PHP语法基础

在了解了如何编写HTML后，那么现在就开始介绍PHP脚本。创建第一个PHP页面，我们完全可以像创建HTML文档那样从头开始做起。请记住下面这个至关重要的事项：*PHP是一种服务器端技术*，这意味着它不能在客户端运行，也就是说不能在Web浏览器中运行。但是Web浏览器能够理解HTML（以及JavaScript和CSS），因此PHP是用来生成在Web浏览器中运行的HTML（请回顾图i-4中对于这种关系的图示）。

标准HTML文档和PHP文档之间有3个主要的不同之处。首先，保存PHP脚本时必须使用`.php`扩展名（例如，`index.php`）。其次，必须把PHP代码放置在`<?php`和`?>`标签之间：

```
...
<body><h1>This is HTML.</h1>
<?php PHP code! ?>
<p>More HTML</p>
...
```

PHP标签指明页面中的这一部分将由PHP执行。这将引出本书要介绍的第3个主要的不同之处：PHP脚本必须运行在启用了PHP的Web服务器上（而HTML页面可以在任何计算机中浏览）。这就是说*PHP脚本必须总是通过一个URL*（如http://*something*/page.php）*运行*。如果在Web浏览器中查看PHP脚本，地址前应该加上http，否则PHP脚本无法工作。

为了让第一个PHP脚本做一些事情而又不被太多编程过程所烦恼，这里将使用`phpinfo()`函数。在调用这个函数时，会向Web浏览器发送一个信息表。这个表列举了安装在特定服务器上的PHP的具体细节。这是检查PHP安装情况的一种非常好的方法，而且具有物超所值的特质。

⇛在本地计算机上创建新的PHP脚本

(1) 在文本编辑器或者IDE中创建一个新的HTML文档，命名为`phpinfo.php`（参看脚本1-2）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type"  content="text/html;   charset=utf-8" />
  <title>First PHP Script</title>
```

```
</head>
<body>
</body>
</html>
```

脚本1-2　第一个PHP脚本使用一个典型的HTML页面，添加PHP标签，并且使用了一个PHP函数

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML1.0 Transitional//EN"
2       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>First PHP Script</title>
7   </head>
8   <body>
9   <?php
10  phpinfo();
11  ?>
12  </body>
13  </html>
```

这段特殊的代码在很大程度上同创建PHP页面没有关系，但是出于一致性的原因，它使用了和基础XHTML示例一样的模板（参看脚本1-1）。

(2) 按Return键（苹果机）或Enter键（PC）以在开启和关闭body标签之间创建一些空白行。

(3) 在开启body标签后的那行输入`<?php`。

这个初始PHP标签告诉服务器接下来的代码是PHP，并且要按照PHP进行处理。

(4) 在下一行添加下面的内容：

```
phpinfo();
```

本书随后将解释该语法的细节，简言之，这是调用名为phpinfo的已有PHP函数。

必须加上一对空的括号“()”，后面接一个分号“;”。

(5) 在关闭body标签之前的那行输入`?>`。

关闭PHP标签告知服务器，脚本中所包含的PHP部分已经结束。任何在PHP标签之外的文本都将立即作为HTML发送给Web浏览器，并且不会被当作PHP代码来处理。

(6) 将脚本保存为`phpinfo.php`。

不想过分强调这一点，但是请记住PHP脚本必须使用有效的文件扩展名。将文件保存为*filename*.php将不会带来任何问题。你还需要确认的是，应用程序或操作系统没有为文件添加隐藏的文件扩展名。例如，Windows的记事本会为不常见的文件扩展名添加.txt，这将使PHP脚本无法使用。

✔**提示**

- ❑ 正如计算机上的文件扩展名会告诉操作系统使用哪种应用程序来打开这种文件一样，一个Web页面的扩展名将告诉服务器如何处理相应的文件：*file*.php通过PHP模块来处理，ASP.NET负责处理*file*.aspx，*file*.html则是静态HTML文档（通常情况下）。这由Web服务器的应用程序设置而决定。

- 如果是为托管的Web网站开发PHP脚本，请向托管公司查询以确认可以为PHP文档使用哪种文件扩展名。在这本书中.php将是最常见的扩展名。
- 读者可能会在别人的脚本中偶尔看到PHP的简写标签<?和?>，但是最好坚持使用正规的标签，正如此书中所示例的那样。实际上，简写标签已在PHP中被废弃了。
- 为phpinfo.php文件创建一份副本会很方便。phpinfo.php脚本会显示出当前PHP版本支持的功能、当前设置及服务器的其他功能。为了查看这些信息，本书后面还会时不时地提醒你运行这个脚本。
- 不使用Web浏览器，也可以执行PHP脚本，但这需要使用命令行界面和一个独立的PHP可执行文件。这个主题超过了本书讨论的范围，而且这种方法也不常用。

1.3　使用 FTP

与HTML能够在自己的计算机上进行测试不同，PHP脚本需要在启用了PHP的服务器上运行以查看输出结果。PHP通过Web服务器应用程序来运行。例如，Apache（http://httpd.apache.org）、Abyss（www.aprelium.com）或者IIS（Internet Information Server，www.iis.net）。

有两种获取PHP服务器的方法：

(1) 在自己的计算机上安装PHP软件；

(2) 申请网络托管主机。

PHP是一个开源软件（从某种程度上说，它是免费的），安装很容易，也不会对计算机产生任何负面影响。如果需要安装PHP以及Web服务器，请参考附录A。如果已经完成安装，你可以跳过本节，直接阅读1.4节，了解如何测试你的第一个PHP脚本。

如果没有在本机运行PHP，那么需要使用FTP软件将PHP脚本传送至启用了PHP的服务器上。Web托管公司或者服务器的管理员将提供使用FTP客户端进行访问的方式。有很多FTP客户端可供选择，在下面的一系列步骤中，本书将选用免费的FileZilla（http://filezilla-project.org/），它可以运行在许多种操作系统中。

⇒使用FTP传送脚本

(1) 打开FTP应用程序。

(2) 在应用程序的连接窗口，输入Web托管公司所提供的信息（参见图1-4）。FTP访问需要主机名或IP地址、用户名和密码。

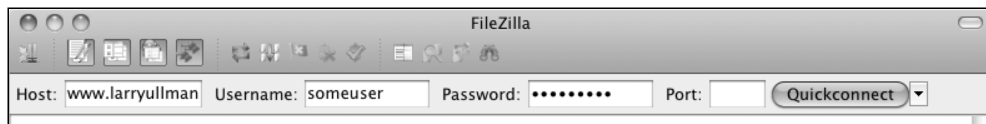


图1-4　在Macintosh中显示的FileZilla应用程序主窗口

(3) 单击快速连接（或者在你的FTP客户端中进行相应的操作）。

如果提供了正确的信息，那么就可以进行连接。否则，将在FileZilla窗口的顶部看到错误消息（参见图1-5）。

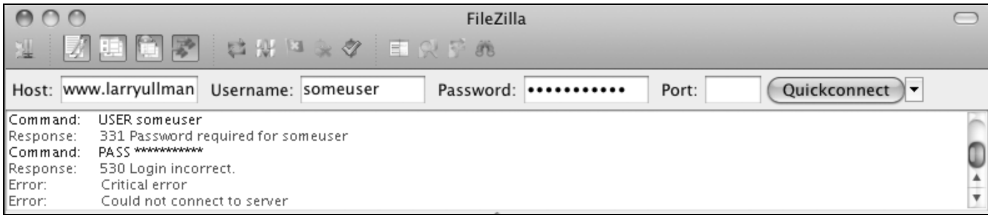


图1-5 这个错误报告表明登录信息不正确

(4) 导航至Web页面的正确目录（例如，`www/`或者`htdocs/`）。

FTP应用程序并不会导航至正确的目录下，这里需要自己导航至Web文档根目录中，这个目录是URL（例如，www.LarryUllman.com）所指向的服务器上的一个位置。如果你不清楚自己电脑的Web文档根目录，可以查看主机托管商提供的说明文档，或向他们寻求帮助。

在FileZilla中，右栏显示位于服务器上的文件和目录，左栏显示位于本机上的文件和目录（参见图1-6）。双击打开文件夹。

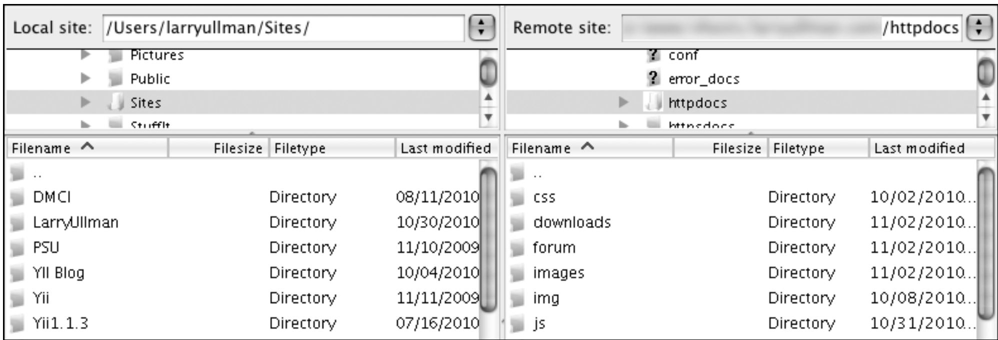


图1-6 我已经成功地同远程服务器建立了连接，并且导航至`httpdocs`目录（Web文档根目录）

(5) 将脚本`phpinfo.php`上传至服务器上。

使用FileZilla上传脚本，只用将文件从左栏中拖曳至右栏中——从本机上传至服务器上。

✓提示

- ❑ 一些文本编辑器和IDE具有内置FTP功能，可以直接将脚本保存到服务器上。比如，Dreamweaver和TextMate不用退出应用就可以运行PHP脚本。

1.4 测试脚本

测试脚本分为两步。第一步，将PHP脚本放在Web服务器的正确目录下。第二步，在Web浏

览器上加载正确的URL运行PHP脚本。

如果使用的是独立的Web服务器（如托管公司提供的主机），只需要使用FTP应用程序上传PHP脚本即可。如果已经在自己的计算机上安装了PHP，那么就可以将PHP脚本保存在（或者移动至）Web文档的根目录中来测试脚本。Web文档的根目录通常表现如下：

- `~/Sites`， Mac OS X用户（~代表主目录）；
- `AbyssDir/htdocs`，在所有操作系统中，AbyssDir是Abyss Web服务器的安装目录；
- `C:\Inetpub\wwwroot`，运行IIS的Windows用户；
- `C:\xampp\htdocs`，运行XAMPP（www.apachefriends.com）的Windows用户；
- `/Applications/MAMP/htdocs`，运行MAMP（www.mamp.info）的Mac用户。

如果你不知道Web文档根目录是什么，查看Web服务器应用程序文档或操作系统文档（如果操作系统内置Web服务应用程序）。

将PHP脚本放到正确的位置后，就可以用浏览器执行该脚本了。

⇛在浏览器中测试脚本

(1) 打开喜爱的Web浏览器。

大多数情况下，PHP在不同的浏览器中表现相同（因为PHP是在服务器上运行的），因此可以随意使用你所喜爱的浏览器。在本书中，将主要使用Firefox和Safari对操作系统没有任何限制。

(2) 在浏览器的地址栏中，输入存放脚本的网站URL。

在本书的示例中将使用www.larryullman.com，但是读者的URL必定不同。

如果在本机运行PHP，那么URL将是http://localhost（在Windows系统中）或http://localhost/~username（在Mac OS X中），请将*username*替换为所使用的真实用户名。一些多合一程序包（如MAMP和XAMPP）还需要在URL中加入端口号，例如：http://localhost:8888。

如果不清楚要使用的URL，可以查看Web服务器应用程序文档。

(3) 向URL中添加`/phpinfo.php`。

如果你将脚本放在Web文档根目录的子目录下，还应该在URL中加上子目录名（如`/ch01/phpinfo.php`）。

(4) 按回车以加载这个URL。

浏览器窗口将加载这个页面（参见图1-7）。

如果看到PHP代码（参见图1-8）或者空白页，可能是由于以下几种情况造成的：

- 没有正确使用ULR加载PHP脚本（如地址名前面漏掉了http）；
- 服务器上没有启用PHP；
- 没有使用正确的扩展名。

如果出现file not found或类似错误（参见图1-9），可能是由于：

- 输入的URL不正确；
- PHP脚本没在放在正确的目录下；
- PHP脚本的名字或扩展名不正确。

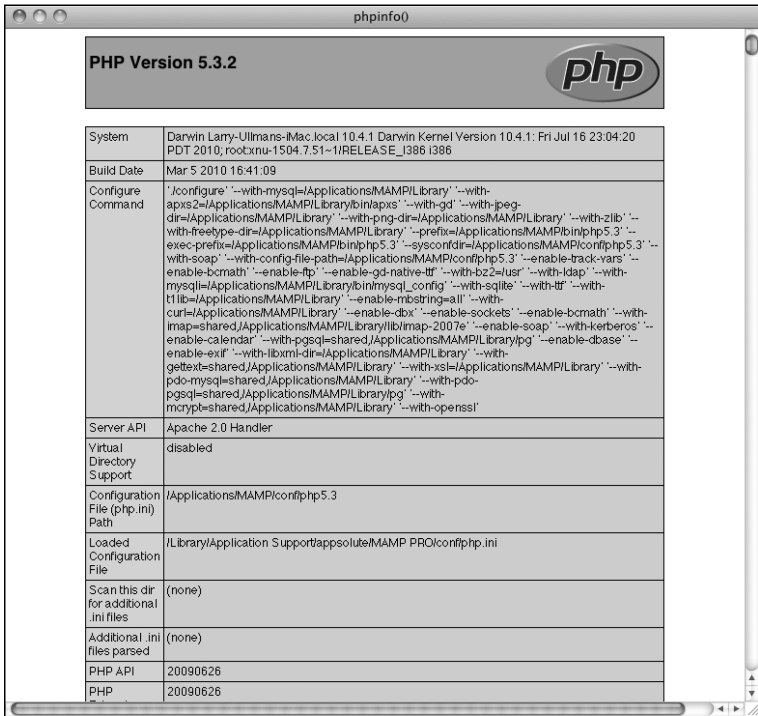


图1-7　如果脚本执行正确，浏览器将会如图所示



图1-8　如果看到PHP源代码，说明没有执行代码



图1-9　服务器的响应说明服务器上没有找到URL访问的文件

✓提示

- ❑ 不能像用其他应用程序打开HTML页面或其他文件那样直接使用浏览器打开PHP文件，这点非常重要！PHP脚本必须使用Web服务器来进行处理，这意味着必须通过一个URL来进行访问（以http://开头的地址）。

- 即使不是一个资深的计算机专家，也应该考虑在自己的计算机上安装PHP。这么做并不困难，而且PHP是免费的。同样，详情请参考附录A。
- 从技术上说，并不需要向phpinfo()脚本中添加任何HTML代码。如果不添加，phpinfo()函数将会生成完整的HTML页面。

1.5　向浏览器发送文本

如果只能知道服务器是不是支持PHP，PHP将显得不那么有用。使用PHP进行的最频繁的操作是以纯文本和HTML标签的方式向浏览器发送信息。比如使用print：

```
print "something";
```

直接输入print，后面添加希望显示的内容，例如一个简短的消息、变量的值、计算的结果等。上述函数中的参数是要打印文本。因为这个参数是文本字符串，所以必须将它们用引号引起来（相对而言，数字则不需要使用引号）。

需要说明的是，print并不会真正*打印*出什么东西，它只是输出数据。当使用Web浏览器运行PHP脚本时，PHP输出的数据只会显示在浏览器上。

也请注意在行结尾处有一个分号（;）。每个PHP代码中的语句必须使用分号作为结尾，忘记这个必要条件是发生错误的常见原因。PHP中的*语句*是一行可执行的代码，例如：

```
print "something";
```

或

```
phpinfo();
```

相反，在本书后面讨论的行注释、PHP标签、控制结构（条件、循环等）以及某些其他结构都不应该使用分号。

最后，你应该了解一些辅助技术细节：print不是真正的函数，它是一种语言结构，而phpinfo()是真正的函数。虽然把print当作函数也是合理的，但print是一种语言结构，在使用时不需要加括号，如上面的例子所示。

⇒ 打印一条简单的消息

(1) 在文本编辑器或者IDE中新建一个HTML文档，命名为hello.php（参看脚本1-3）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Hello, World!</title>
</head>
<body>
<p>The following was created by PHP:
```

上述代码基本上全是标准HTML。最后一行用来区别硬编码的HTML和由PHP生成的HTML。

脚本1-3　通过在PHP标签间放置print语句，服务器将动态地向浏览器发送Hello, world!问候

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
6     <title>Hello, World!</title>
7   </head>
8   <body>
9   <p>The following was created by PHP:
10  <?php
11  print "Hello, world!";
12  ?>
13  </p>
14  </body>
15  </html>
```

(2) 在下面新的一行，输入<?php来创建PHP起始标签。

(3) 添加：

```
print "Hello, world!";
```

打印短语Hello, world!是大多数编程参考书目所教授的第一步。尽管这对于使用PHP来说是微不足道的理由，但是如果还没有编写过至少一个Hello, world!应用程序的话，你将不能称之为一个真正的程序员。

(4) 关闭PHP部分以及HTML页面：

```
?>
</p>
</body>
</html>
```

(5) 将文件保存为hello.php，放在启用了PHP的服务器上，然后在浏览器中测试（参见图1-10）。

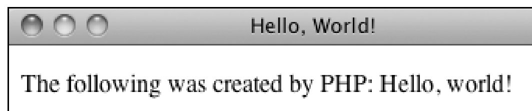


图1-10　一个简单的Hello, world!示例：第一个PHP编程尝试

如果在自己的计算机上运行PHP，请记住将文件保存在适当的目录下，并且通过访问http://localhost/来访问脚本。

如果你看到错误或者空白页面而不是如图所示的正确结果，请查看本章最后的调试部分。

✓提示

- PHP是不区分大小写的，因此当通过print、Print和PRINT调用print或phpinfo()函

数时结果都是一样的。在本书后面的部分中（如第2章）将看到大小写起重要作用的示例。

- 可以使用其他函数（例如，echo和printf()）向浏览器发送文本，但是在本书中，将主要使用print。
- 通常情况下可以使用print函数作用于多行文本：

```
print "This is a longer sentence of text.";
```

以右引号（"）结束要打印的消息，并在语句的末尾加上分号（;）。

1.6　使用 PHP 手册

可以在线访问PHP手册（www.php.net/manual），它列举了在这种语言中可以使用的每一个函数，但是使用手册需要一些专业知识。该手册按照使用PHP的先后顺序进行组织（安装、语法、变量），最后是专题函数（MySQL、字符串函数等）部分。

要迅速地在PHP手册中找到函数，请在Web浏览器中访问www.php.net/*functionname*（例如，www.php.net/print）。

为了理解如何描述函数，请查看print函数页的开头（参见图1-11）：

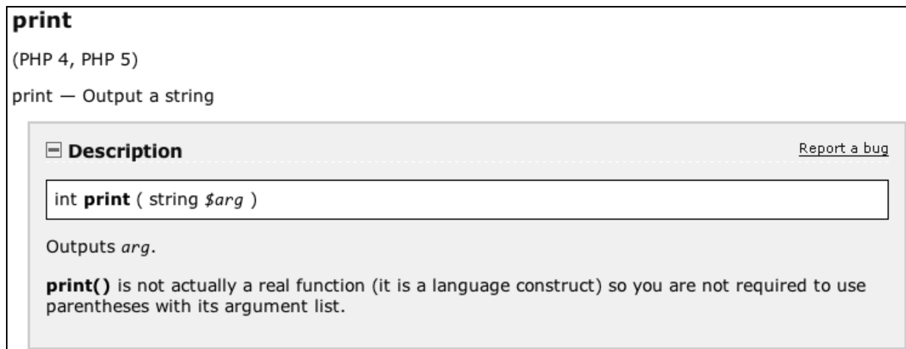


图1-11　PHP手册中的print函数结构

第一行是函数本身的名称，然后是可以使用这个函数的PHP版本。由于语言在发展，会不断加入的新函数，也会偶尔移除旧函数。接下来是对这个函数的文字描述以及函数的基本用法。用法是最重要也是最容易混淆的部分。

在这个示例中，第一个值int表明print返回一个整型值（print正常情况返回1）。在圆括号内string $arg说明该函数必须接受一个参数，而且必须是字符串形式。我们已经实际使用过了。

作为对比，在手册中找到nl2br()（参见图1-12）。这个函数将文本中的换行符（相当于按下Return或Enter键）替换成HTML的换行标签。这个函数返回一个字符串，将一个字符串作为第一个参数，将一个可选的布尔型（TRUE/FALSE）作为第二个参数。当你在手册中看到方括号时，就表示其内的参数是可选参数，这些可选参数会列在最后。如果函数接受多个参数，要使用逗号

将它们分隔开来。你可以这样调用该函数：

```
nl2br("Some text");
nl2br("Some text", false);
```

这个函数的定义还说明第二个参数有一个默认值TRUE，表示如果不给第二个参数传入FALSE值，将默认创建XHTML的`<br/>`标签。在上例中，该函数会创建HTML的`<br>`标签。

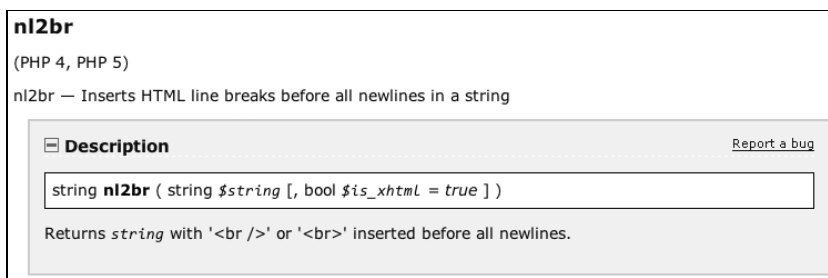


图1-12　PHP手册中的`nl2br()`函数

如果曾经因为函数而迷惑，或者不知道如何正确使用它，请查看PHP手册参考页以寻求帮助。

⇛ 查找一个函数定义

(1) 在你的Web浏览器中输入www.php.net/*functionname*。

如果PHP手册中没有与你输入的函数匹配的记录，查看函数拼写是否正确或看看手册中推荐的替代函数（参见图1-13）。

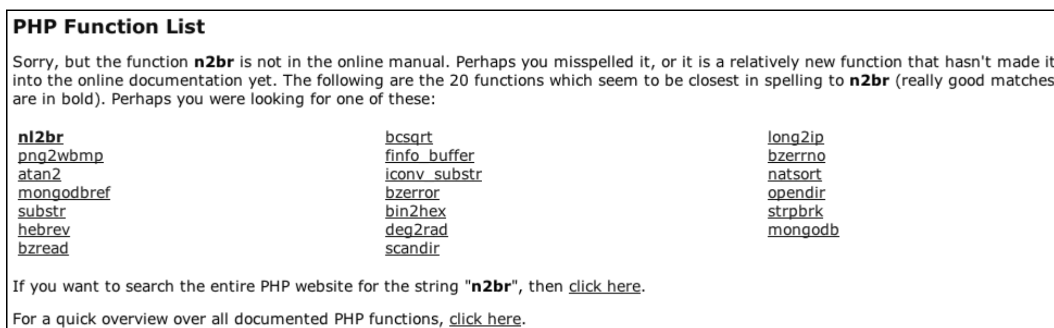


图1-13　如果URL中输入的函数名不完全匹配，PHP手册会列出一些替代函数

(2) 将函数所在的PHP版本与你使用的PHP版本作比较。

使用前面介绍过的`phpinfo()`函数，确认你使用的PHP版本。如果某个函数是在后面的版本中加入的，你可以升级PHP版本，也可以找其他方法实现。

(3) 检查函数返回的数据类型。

有时会因为函数返回值而出现问题，比如它返回的类型并不是你想要的。

(4) 检查函数需要（或接受）的参数数量和参数类型。

使用函数的常见错误是，在调用函数时传入错误的参数数量和参数类型。

(5) 如果有用户注释，请多加关注，你会从中学到更多知识。

用户注释有时非常有用（有时没什么用）。

✓提示

- 如果你看到一条警告消息，指出某个函数已经过时（参见图1-14），就表示这个函数会在未来的PHP版本中移除，你应该使用更新的、更好的替代函数（可以确定，一定会有一个函数替代它）。

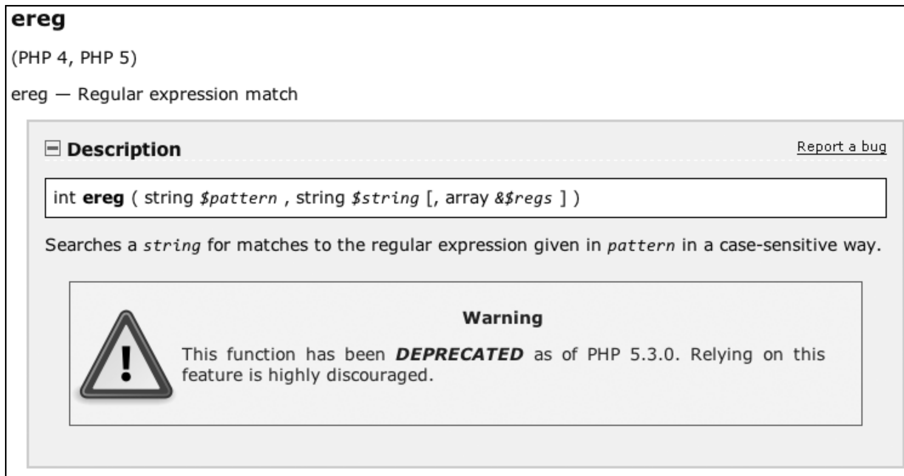


图1-14　在代码中应该避免使用过时的函数

1.7　向浏览器发送 HTML

首次学习HTML的人都会很快发现，在Web浏览器中查看纯文本还有很多不足之处。确实，创造HTML的初衷就是为了让纯文本更加吸引人而且有用。由于HTML通过向文本添加标签来工作，因此可以使用PHP向浏览器发送HTML标签以及其他数据：

```
print "<b>Hello, world!</b>";
```

这里有一种情况需要留意，对于需要使用引号的HTML标签(如link ）将在使用PHP进行打印时发生问题，因为print函数也使用引号（参见图1-15)：

```
print "<a href="page.php">link</a>";
```

解决方法是通过在引号前面加上反斜线（\）的方式将其转义：

```
print "<a href=\"page.php\">link</a>";
```

通过将print语句中的引号进行转义，PHP会打印字符本身，而不是将其解释为需要打印的字符串的起始和结尾。

Parse error: syntax error, unexpected T_STRING in **/Users/larryullman/Sites/phpvqs4/hello.php** on line **11**

图1-15 尝试打印双引号会出现错误提示，因为print语句中也要使用双引号

⇛ 向浏览器发送HTML的步骤

(1) 在文本编辑器或IDE中打开hello.php脚本（参看脚本1-3）。

(2) 编辑第11行中Hello, world!文本，添加HTML标签，如下所示（参看脚本1-4）：

```
print "<span style=\"font-weight: bold;\">Hello, world!</span>";
```

脚本1-4 使用print函数，可以将HTML标签和文本一起发往浏览器，浏览器会根据标签对文本进行格式化

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>Hello, World!</title>
7   </head>
8   <body>
9   <p>The following was created by PHP:
10  <?php
11  print "<span style=\"font-weight: bold;\">Hello, world!</span>";
12  ?>
13  </p>
14  </body>
15  </html>
```

让消息中PHP生成的部分突出显示，我们应用了一些CSS，用粗体显示它们。要达到这个目的，需要在span标签中使用反斜线来转义引号，以避免和print语句中的引号发生冲突。

(3) 将脚本保存为hello2.php，放置在启用了PHP的服务器上，并且在浏览器上运行该页（参见图1-16）。



图1-16 页面的新版本有了一定的装饰和吸引力

(4) 查看HTML页的源代码以查看发送到浏览器的代码（参见图1-17）。

怎样做将取决于所使用的浏览器：在Firefox中选择View>Page Source，在IE中为View>Source，在Safari中是View>View Source。



图1-17　Hello2.php页中HTML源文件的显示结果（参看脚本1-4）

有必要将这一步作为习惯操作，尤其是当有错误发生时。请记住PHP主要用来生成HTML，然后向Web浏览器发送并通过浏览器解释它们。通常确认向Web浏览器发送的内容（用查看源文件的方式）将有助于发现在浏览器解释过程中出现的问题。

✓提示

- 对于PHP程序员来说，理解引号的作用以及如何对有问题的字符进行转义至关重要。接下来的两章将更详细地介绍这些主题。
- 通过PHP向Web浏览器发送的HTML不必如此简单。还可以创建表格、JavaScript以及更多的内容。
- 请记住，任何在PHP标签之外的HTML都将自动发送至浏览器。在PHP标签中，print语句被用来向浏览器发送HTML。

使用空白

当使用PHP进行编程时，应该理解空白通常会被忽略，但也有例外。空白行（一行或者几行）并不影响最后的结果。同样，制表符和空格对PHP来说通常无关紧要。由于PHP代码在Web浏览器中并不可见（除非服务器出了问题），因此在PHP文件中的空白并不会对最终用户查看页面结果造成影响。

显示在Web页面HTML源代码中的空格，对在浏览器中查看页面产生的影响极小。例如，将所有的源代码放置在一行中将不会改变其实际效果。但是，如果需要在HTML源代码中查找问题，你一定不会希望所有HTML代码都出现在一行上。你可以通过PHP控制多行输出来解决空格和换行问题，或者在两个双引号之间加上换行符（\n）将HTML代码分行。

```
print "Line 1\nLine 2";
```

使用换行符只会影响HTML的源代码，而不会呈现在最终用户的浏览器中。

如果要改变浏览器中呈现的空白，需要使用CSS、以及<p>、<div>和换行标签等。

1.8　为脚本添加注释

注释是对编程的完善，这并不是因为它可以做什么，而是因为它能够帮助程序员今后想起为什么这么做。计算机在处理脚本的时候将忽略这些注释。此外，PHP注释将不会被发送至Web浏览器，因此能够保守秘密。

PHP提供3种添加注释的方式。可以在希望不运行的单行代码开头使用//或者#将其注释掉。例如：

```
// 这是一条注释。
```

还可以使用//或者#在PHP语句的结尾添加注释，例如：

```
print "Hello"; // 只是一句问候。
```

虽然是一个使用习惯问题，但在PHP中//比#更常用。

还可以在开头和结尾使用/*和*/注释多行：

```
/* 这是
多行注释。*/
```

一些程序员喜爱这种注释风格，因为它包括开始和结束“标签”，为注释划分了明显的开始点和结束点。

⇛为脚本添加注释

(1) 在文本编辑器或IDE中打开hello2.php（参看脚本1-4）。

(2) 在PHP起始标签后为脚本添加一些注释（参看脚本1-5）：

```
/*
 *  文件名：hello3.php
 *  书中引用：脚本1-5
 *  由Larry Ullman创建
 */
```

脚本1-5　在一行代码前放置//或#，将导致PHP不再对该行进行处理

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>Hello, World!</title>
7   </head>
8   <body>
9   <p>The following was created by PHP: <br/>
10  <?php
11  /*
12   * 文件名：hello3.php
13   * 书中引用：脚本1-5
14   * 由Larry Ullman创建
15   */
```

```
16
17  //print "<span style=\"font-weight: bold;\">Hello, world!</span>\n";
18
19  ?>
20  <!-- 这是一个HTML注释。-->
21  </p>
22  </body>
23  </html>
```

这只是能编写的多种注释中的一个示例。本书强烈建议读者使用注释记录下脚本的功能，它依赖什么信息，由谁创建，什么时候创建等信息。看样式就知道，这种注释一般都放置在脚本的顶部（作为PHP代码段的开头）。额外的星号并不是必需的，它们只是强调这是一个注释。

(3) 在第17行的`print`语句之前输入`//`。

在`print`语句之前添加`//`可以用来将这个函数注释掉，这就意味着它将不会被执行。

(4) 在关闭PHP标签之后（第19行），添加一个HTML注释：

```
<!-- 这是一个HTML注释。-->
```

这行代码将有助于你理解不同的注释以及它们在哪里出现。这个注释将只在HTML源代码中出现。

(5) 将脚本保存为`hello3.php`，放置在启用了PHP的服务器上，然后在Web浏览器上运行该页（参见图1-18）。

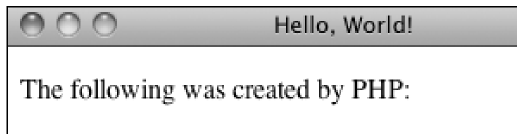


图1-18　当`print`语句被注释掉后，页面看起来如同`print`函数根本没有出现在代码中一样

(6) 查看该页的源代码以了解HTML注释（参见图1-19）。




```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Hello, World!</title>
</head>
<body>
<p>The following was created by PHP: <br />
<!-- This is an HTML comment. -->
</p>
</body>
</html>
```

图1-19　HTML注释不会出现在Web浏览器中，但是会存在于HTML源代码里。PHP的注释将只出现在服务器上的PHP脚本中

✓提示

- ❑ 可以使用/*和*/将单行代码或者多行代码注释掉。//或者#一次只能注释掉一行代码。
- ❑ 不同的程序员偏爱不同的代码注释方式。最重要的是找到一种对自己适用的方式，并且坚持使用。
- ❑ 请注意，不能在PHP代码中使用HTML注释符号（<!--和-->）来注释代码。可以使用PHP向浏览器中打印这些标签，但是在这种情况下，你就已经创建一个注释，因为它出现在客户端计算机上的HTML源代码中（而不是浏览器窗口中）。而PHP注释永远不会出现在用户的计算机上。
- ❑ 尽管我十分坚信你不会过度注释代码，但是在本书中将不会注释所有脚本，这是出于节省纸面空间的考虑。但是本书将对脚本的名称及编号给出注释，以达到对照参考的目的。
- ❑ 当你修改脚本代码时，不要忘记同时更新注释。如果注释中的说明与脚本、代码行的行为相悖，会让人困惑。

1.9 调试的基本步骤

调试不是一个可以简单掌握的概念，它是通过实际动手才能真正掌握的一门技术。即便是接下来的50页专门针对这个主题进行阐述，恐怕你也只能掌握必备调试技能的一小部分。

本书在这里如此介绍调试的原因是想让你记住，在想清楚之前不要匆忙地去编程。有时代码并没有像期望的那样工作，而程序员不可避免地会造成疏漏和错误，这会令人痛苦并抓狂，即便是使用像PHP这样友好的语言也是如此。我从1999年开始使用PHP编程，但是偶尔还是会陷入编程的泥沼中，因此调试是需要掌握的重要技能，读者将最终认识到它的必要性并深深体会到这一点。作为PHP编程历险的开始，本书将提供下面这些基础但又具体的调试技巧。

⇛ 调试PHP脚本

- ❑ 确认总是通过URL来运行PHP脚本！这可能是初学者常犯的错误。PHP代码必须通过Web服务器应用程序来运行，也就是说必须通过http://*something*发出请求。如果你看到的是PHP代码，而不是运行结果的话，就说明PHP脚本并没有通过URL执行。
- ❑ 熟知运行的PHP版本。一些问题是由PHP的版本所致。在使用启用PHP的服务器之前，运行`phpinfo.php`文件（参看脚本1-2）来确定使用的PHP版本。
- ❑ 确保`display_errors`是开启状态。这是一个PHP基础配置设置（在附录A中有相关的讨论）。要确认这些配置，可以执行`phpinfo()`函数（使用浏览器，在结果页中搜索`display_errors`）。出于安全的考虑，PHP可能没有设置显示所发生的错误。如果是这样，将在问题发生时只能看到空白的页面。为了对该问题进行调试，需要查看这些错误，因此在学习的时候请将这个设置打开。可以在附录A和第3章中找到相关的介绍。

- 检查HTML源代码。有时问题隐藏在页面的HTML源代码中。事实上，有时PHP错误消息正是隐藏在那里的！
- 相信错误消息！初学者常犯的另外一个错误是不完整阅读PHP报告的错误消息，或者不完全相信这些错误。尽管这些错误消息有时非常含糊，并且看上去没有什么意义，但是还是不能忽略它！至少在错误发生于哪行代码这个问题上，PHP通常都是正确的。如果需要将错误信息转发给别人（比如你请我帮你分析出错原因时），一定注意要包括全部错误信息。
- 休息一下！这么多年来我遇到过很多的编程问题，并且绝大多数最困难的问题，都是在我离开计算机去散步一会回来后解决的。编程时很容易陷入挫败与迷惑交织的痛苦中，在这种情况下，所采取的任何进一步措施似乎都会将事情弄得更糟。

✓提示

- 这是一些常见的调试技巧，适用于PHP编程初学者。对于现在来说应该已经足够，因为本书中的相关示例都非常简单。代码越复杂，对调试技术的要求也就越高，因此我编著的《PHP 6与MySQL 5基础教程》专门用一章来讨论这个主题。

1.10　回顾和实践

本书会在每章的最后添加“回顾和实践”一节。在本节中，我会就本章涉及的内容提出一些问题，目的是巩固和扩展你学到的知识和技能。无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书的论坛上（www.LarryUllman.com/forum/），我们会为你答疑解惑。

1.10.1　回顾

- HTML是什么？XHTML是什么？这两者之间有什么不同之处？
- 你的文本编程器或IDE使用的是哪种编码？它与你在HTML代码中指定的相同吗？
- CSS是什么，用来做什么？
- 你服务器上PHP脚本的扩展名是什么？
- 什么是“Web根目录”？你的服务器的Web根目录是什么？
- 如何测试PHP脚本？如果PHP脚本没有运行，是什么原因？
- 说出两个在PHP代码中添加注释的方法。说说注释对你都有哪些用处。

1.10.2　实践

- 如果需要访问多个服务器，确认其他服务器上运行的PHP版本。
- 创建一个静态HTML网页，显示一些信息。用PHP生成一些内容替换网页上部分静态信息。

- 创建一个模板，该模板必须包含HTML的基本框架、PHP开始标签和结束标签以及一些基本注释信息。
- 使用phpinfo()函数确认服务器上已启用display_errors。如果未启用，更改服务器配置来启用它（详细内容见第3章和附录A）。
- 后面章节会用到一些新的函数，你应该随时查看PHP手册。

第 2 章

变　　量

2

本章内容

- ❑ 什么是变量
- ❑ 变量语法
- ❑ 变量类型
- ❑ 为变量赋值
- ❑ 理解引号
- ❑ 回顾和实践

在第1章中，已经使用PHP向Web浏览器发送了简单的文本和HTML代码——换句话说，做这些事其实根本用不着PHP！别担心，本书将教授你如何联合使用`print`和其他的PHP特性来为Web网站做有用的事。

为了实现从创建简单的静态页面向动态Web应用程序和交互式Web网站的飞跃，需要使用变量。同在其他编程语言中一样，变量也是PHP中的一个核心概念。理解什么是变量，一门语言支持什么类型的变量，以及如何使用它们，对你的工作至关重要。

本章将讨论在PHP中所使用的变量的基础知识，后续章节则会详细介绍各种变量。如果从未接触过变量，本章对读者来说将是很好的入门。如果已经深谙此道，在本章的学习中读者将轻车熟路。

2.1　什么是变量

最好的理解是把*变量*想象为存储数据的容器。一旦数据被存储在一个变量中（更加精确地说，一旦变量被赋值），数据/变量将能够被改变，在Web浏览器上打印出来，存储在数据库中、被电子邮件发送，等等。

PHP中的变量本质上是灵活多变的：可以为一个变量赋值、在变量中检索数据（而不影响变量的值）、为变量赋新的值，以及在需要的时候进行循环。但是PHP中的变量也是临时的：*它们只存在于一个脚本的执行期间*。当脚本的最后一个PHP标签被执行过后，这些变量也就不存在了。当用户单击一个链接或者提交表单时，将获得一个新的页面，同时这些变量也将不复存在。

在深入探讨变量之前，我们写一个简单的脚本，用来列出一些PHP中的预定义变量。这些变量不需创建即可使用，这是因为PHP已经为用户完成了创建工作。在本书的课程中，会介绍许多不同的预定义变量。在这个示例中，使用了$_SERVER这个预定义变量。它包含了关于运行PHP的计算机的许多信息。

print_r()函数提供了一种显示变量值的简单方式：

```
print_r($variable_name);
```

只要将要查看的变量名作为print_r()的参数就可以显示变量的值(在这一章中你会学到很多关于变量的语法)。

⇛ 打印出PHP预定义的变量

(1) 在文本编辑器或者IDE中创建新的PHP脚本，命名为predefined.php（参看脚本2-1)。

脚本2-1 调用print_r()函数以便查看储存在$_SERVER变量中的值

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Predefined Variables</title>
7   </head>
8   <body>
9   <pre>
10  <?php // 脚本2-1 - predefined.php
11
12  // 显示 $_SERVER 变量的值:
13  print_r ($_SERVER);
14
15  ?>
16  </pre>
17  </body>
18  </html>
```

(2) 创建HTML初始标签：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Predefined Variables</title>
</head>
<body>
<pre>
```

这段代码复用了在第一章中创建的XHTML模板。在页面的body部分，本书使用<pre>标签让生成的PHP信息更加清晰易读。尽管不赞成使用这些标签（我的建议是再也不要使用它们)，但是它们能够很好地达到这个目的。如果不使用<pre>标签，由print_r()函数生成的结果将显

得非常凌乱。

(3) 添加PHP代码：

```
<?php // 脚本2-1 - predefined.php print_r($_SERVER); ?>
```

这段PHP代码只是用来调用`print_r()`函数。必须为该函数提供变量名。例如，变量名为`$_SERVER`，这是PHP中一个特殊的变量。`$_SERVER`变量储存关于服务器的所有数据：名称、操作系统、当前用户名、Web服务器应用程序（Apache、Abyss、IIS等）的信息，以及其他信息。它也提供所执行的PHP脚本的信息：脚本名称、在服务器上保存的位置等。

请注意，必须严格按照变量全大写的样式正确输入`$_SERVER`。

(4) 完成HTML页面：

```
</pre>
</body>
</html>
```

(5) 将文件保存为`predefined.php`并上传至服务器（或者保存在本机的目录下），并且在Web浏览器上测试（参见图2-1）。再次强调，请记住必须通过URL来运行所有的PHP脚本。

```
Array
(
    [HTTP_HOST] => phpvqs4:8888
    [HTTP_USER_AGENT] => Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6;
    [HTTP_ACCEPT] => text/html,application/xhtml+xml,application/xml;q=0
    [HTTP_ACCEPT_LANGUAGE] => en-us,en;q=0.5
    [HTTP_ACCEPT_ENCODING] => gzip,deflate
    [HTTP_ACCEPT_CHARSET] => ISO-8859-1,utf-8;q=0.7,*;q=0.7
    [HTTP_KEEP_ALIVE] => 115
    [HTTP_CONNECTION] => keep-alive
    [PATH] => /usr/bin:/bin:/usr/sbin:/sbin
    [SERVER_SIGNATURE] =>
    [SERVER_SOFTWARE] => Apache
    [SERVER_NAME] => phpvqs4
    [SERVER_ADDR] => 127.0.0.1
    [SERVER_PORT] => 8888
    [REMOTE_ADDR] => 127.0.0.1
    [DOCUMENT_ROOT] => /Users/larryullman/Sites/phpvqs4
    [SERVER_ADMIN] => you@example.com
    [SCRIPT_FILENAME] => /Users/larryullman/Sites/phpvqs4/predefined.php
    [REMOTE_PORT] => 51149
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /predefined.php
    [SCRIPT_NAME] => /predefined.php
    [PHP_SELF] => /predefined.php
    [REQUEST_TIME] => 1289831848
    [argv] => Array
        (
        )

    [argc] => 0
)
```

图2-1　该脚本打印了`$_SERVER`变量的值，它是有关服务器和PHP脚本的值的主列表

(6) 如果有可能，将文件保存在另外一台可运行PHP的计算机或者服务器上，然后在Web浏览器上重新运行脚本（参见图2-2）。

```
Array
(
    [HTTP_HOST] => larryullman.com
    [HTTP_USER_AGENT] => Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_5; en-us
    [HTTP_ACCEPT] => application/xml,application/xhtml+xml,text/html;q=0.9,text/
    [HTTP_ACCEPT_LANGUAGE] => en-us
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_CONNECTION] => keep-alive
    [PATH] => /sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin
    [SERVER_SIGNATURE] =>
Apache/2.0.63 (CentOS) Server at larryullman.com Port 80


    [SERVER_SOFTWARE] => Apache/2.0.63 (CentOS)
    [SERVER_NAME] => larryullman.com
    [SERVER_ADDR] => 207.58.187.78
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => 71.58.97.51
    [DOCUMENT_ROOT] => httpdocs
    [SERVER_ADMIN] => Larry@DMCInsights.com
    [SCRIPT_FILENAME] => httpdocs/predefined.php
    [REMOTE_PORT] => 44766
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /predefined.php
    [SCRIPT_NAME] => /predefined.php
    [PHP_SELF] => /predefined.php
    [REQUEST_TIME] => 1289832083
)
```

图2-2　不同的服务器在运行predefined.php时，页面将显示不同的结果（请同图2-1进行对比）

✓提示

- ❑ 像前面示例中这样打印出变量的值是进行调试的主要手段，这是因为通常问题出在变量中没有保存想要的值。脚本有时不会按预期工作，因为某些变量的值可能和你想的不一样，因此确认变量的实际值非常重要。
- ❑ 如果不使用<pre></pre> HTML标签，结果就会如图2-3中显示的那样非常凌乱。

```
Array ( [HTTP_HOST] => phpvqs4:8888 [HTTP_USER_AGENT] => Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.2.12) Gecko
/*;q=0.8 [HTTP_ACCEPT_LANGUAGE] => en-us,en;q=0.5 [HTTP_ACCEPT_ENCODING] => gzip,deflate [HTTP_ACCEPT_CHARSET] => I
[HTTP_CACHE_CONTROL] => max-age=0 [PATH] => /usr/bin:/bin:/usr/sbin:/sbin [SERVER_SIGNATURE] => [SERVER_SOFTWARE] => Ap
[REMOTE_ADDR] => 127.0.0.1 [DOCUMENT_ROOT] => /Users/larryullman/Sites/phpvqs4 [SERVER_ADMIN] => you@example.com [SCRIP
[GATEWAY_INTERFACE] => CGI/1.1 [SERVER_PROTOCOL] => HTTP/1.1 [REQUEST_METHOD] => GET [QUERY_STRING] => [REQUE
[REQUEST_TIME] => 1289832176 [argv] => Array ( ) [argc] => 0 )
```

图2-3　HTML预格式化标签能够避免在使用print_r()时出现这样令人难以看清的页面（请同图2-1和图2-2进行对比）

2.2　变量语法

现在已经初步了解了变量，接下来将进一步探讨有关变量的内容。在先前的示例中使用了PHP中$_SERVER这个预定义变量。在掌握了正确的语法后，也可以自己创建变量。创建正确的变量名，必须遵循以下规则：

- ❑ 所有的变量名必须以美元符号（$）开头；
- ❑ 在美元符号后的第一个字符必须为字母（A～Z，a～z）或者下划线（_），不能使用数字；
- ❑ 变量名剩下的部分可以包含任何数量的字母、数字和下划线组合；
- ❑ 变量名中不能出现空格（通常使用下划线进行文字分隔）；
- ❑ 变量名必须唯一；
- ❑ 变量名是区分大小写的！这意味着`$variable`和`$Variable`是截然不同的变量，因此将两个变量进行如此相似的命名是非常不明智的。

最后也是最重要的一点：PHP中的变量名是区分大小写的。使用错误的大小写字母是导致bug发生的常见原因。例如，如果在前面脚本中使用`$_server`或`$_Server`，你可能会收到错误信息或什么也看不到（参见图2-4）。

```
Notice: Undefined variable: _server in /Users/larryullman/Sites/phpvqs4/predefined.php on line 13
```

图2-4　错误地拼写变量名（包括大小写错误），可能出现不需要的或无法预计的结果

本书提供以下建议来帮助读者尽可能减少bug发生的可能性。

- ❑ 变量名全部使用小写字母。
- ❑ 让变量名具有描述性（例如，`$first_name`就比`$fn`好得多）。
- ❑ 使用注释来说明变量的用途（参看脚本2-2），虽然看上去这似乎有些多余。
- ❑ 最重要的，保持一致的命名模式。

表2-1列出了一些有效的变量名。

表2-2列出了一些无效的变量名及其违反的规则。

表2-1　PHP中有效的变量名

变　量　名
$first_name
$person
$address1
$_SERVER

表2-2　PHP中无效的变量名

变　量　名	原　　因
$first name	包含空格
$first.name	包含英文句点
first_name	没有以$开头
$1address	$后的第一个字符不能为数字

脚本2-2　这段代码显示了如何使用注释来说明变量的用途，通常详细地进行注释总是比不用注释要好得多

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4  <head>
5    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6    <title>Variables and Comments</title>
7  </head>
8  <body>
9  <?php // 脚本2-2
```

```
10
11 // 定义变量……
12
13 $year = 2011; // 当前年。
14 $june_avg = 88; // 6月份的平均气温。
15 $page_title = 'Weather Reports'; // 页面的标题。
16
17 // ……等等。
18
19 ?>
20 </body>
21 </html>
```


✓提示

- ❑ 与其他语言不同，PHP通常不需要在使用变量前声明或者初始化变量。换句话说，可以在定义变量类型之前先引用。尽管如此，最好还是不要这样做。本书所编写的脚本将尽力做到先定义变量后再使用。
- ❑ 变量命名的惯例主要有两种，可以根据所使用的不同描述方式来选择。它们是所谓的驼峰（camel-hump）方式（使用大写字母对单词进行分隔，例如`$FirstName`）和下划线方式（例如`$first_name`）。本书将在示例中使用后一种方式。

2.3　变量类型

本书将介绍3种变量类型：数值型、字符串型和数组型。这里只作简要的介绍，后面几章（第4章、第5章和第7章）将对详细介绍它们。第4种变量类型——对象，将在附录B中介绍，而不包含在本书正文中，因为它是一个特殊的主题，对于初学者来说难度太大。事实上，在我编著的*PHP 5 Advanced: Visual QuickPro Guide*（Peachpit 2007年出版）中，使用了超过150页的篇幅来讲述这个主题，却仅仅只涉及了这个主题的基础部分。

2.3.1　数值

从技术上讲，在PHP中数值类型分为两种：整型和浮点型（也被称为双精度浮点型）。因为PHP对于变量的处理方式相对随意，所以不会阻止对两种类型的数值进行同样的处理。同样，为了表述清晰本书将对两种类型的区别作简明介绍。

数值的第一种类型——整型，同整数一样。它们可以是正数或者负数，但是不能包含有分数或者小数部分。带有小数点的数值（就像1.0）是浮点型。必须使用浮点型的数值来表示分数，由于在PHP中表示分数的唯一方式就是将之转换为等价的小数，因此$1\frac{1}{4}$用1.25表示。表2-3列举了一些有效数值的示例以及它们对应的类型，表2-4列举了无效的数值及其违反的规则。

表2-3　PHP中的有效数值

数　值	类　型
1	整型
1.0	浮点型
1972	整型
19.72	浮点型
−1	整型
−1.0	浮点型

表2-4　PHP中的无效数值

数　值	原　因
1/3	包含斜线
1996a	包含字母
08.02.06	包含多个小数点

✓提示

- ❑ 下节就可以了解到，可以将无效的数值用引号引用从而转换为有效的字符串。

2.3.2　字符串

一个字符串是一对单引号（'）或者双引号（"）引用的任意数量的字符。字符串能够包含字母、数字、符号和空格的任何组合，它还能够包含变量。

下面是有效字符串的示例：

```
"Hello, world!"
"Hello, $first_name!"
"1/3"
'Hello, world! How are you today?'
"08.02.06"
"1996"
''
```

最后一个示例是一个*空字符串*：一个没有包含任何字符的字符串。

简言之，创建字符串，就是用引号引用一些字符。但在使用时，你可能会遇到一些问题，例如：

```
"I said, "How are you?""
```

这个示例有些不好理解，本书在第1章中介绍打印HTML代码时提到过这样的问题。当PHP遇到第2个引号时，它认为这是字符串的结束标志，接下来的文本（How及其后的字符）将会引发错误。如前所述，可以在字符串里使用引号，但是需要在其之前放置反斜线（\）将其转义：

```
"I said, \"How are you?\""
```

这样就可以告诉PHP这两个引号是字符串的值，而不是字符串开启和关闭的指示符。

可以使用不同的引号类型来解决这个问题：

```
'I said, "How are you?"'
"I said, 'How are you?'"
```

✓提示

- 请注意`"1996"`这个示例，将整型字符放置在引号中就转换成为字符串型。从本质上说，字符串中包含的字符`1996`同数值`1996`（未被引用）相等。这是一个难以看出的区别，而且在代码中不会引发问题。这是因为可以使用字符串`1996`进行数学运算，就如同使用数值`1996`一样。
- 在第1章中，本书介绍了如何使用被双引号引用的`\n`字符创建一个新行。转义引号将打印引号，转义n将打印新行，转义r可以创建一个回车符，转义t将向HTML源代码中插入一个制表符。
- 理解字符串、变量和单双引号对于使用PHP编程来说非常重要。出于这个原因，本书将在2.5节专门阐述这个主题。

2.3.3 数组

本书将在第7章对数组进行详细的讨论，这里只作简明的介绍。字符串和数值类型（都被称之为*标量*）只包含一个值，数组可以被赋予多个值，可以认为数组是值的列表。换句话说，可以在一个数组中放置多个字符串和/或数值。

数组使用*键*来创建和检索它们保存的值，其结果构成为一系列键-值对，看上去像是一个两列的数据表。有趣的是，在PHP中的数组结构非常灵活，它的键和值都可以使用数值或者字符串类型。数组甚至不必在此方面保持一致性（在第7章接触到具体示例时，你就会明白这里所说的意思）。

PHP有两种数组类型，区别在于键的格式。如果一个数组使用数值作为键（参见表2-5）那么它就是*索引*数组，如果它使用字符串作为键（参见表2-6），那么就是*关联*数组。在任何一种情况下，数组中的值都可以是任何变量类型（字符串、数值或者其他类型）。

✓提示

- 数组的键又被叫做*索引*，这两个术语意思相同。
- 一个数组可以并且经常包含其他的数组，这种数组叫做*多维*数组。
- PHP中所谓的*关联*数组，在Perl和Ruby中叫做*散列*。

表2-5 索引数值

键	值
0	Don
1	Betty
2	Roger
3	Jane

表2-6 关联数值

键	值
VT	Vermont
NH	New Hampshire
IA	Iowa
PA	Pennsylvania

2.4　为变量赋值

在为变量赋值时，可以忽略变量的类型，使用等号（=）给变量赋值。因此，等号被称作为*赋值运算符*，并且变量在赋值运算符左边，而被赋的值在右边。例如：

```
$number = 1;
$floating_number = 1.2;
$string = "Hello, world!";
```

每一行都是一个完整的赋值语句（即一个可执行的动作），每一句结尾都要有一个分号。

打印变量值，可以使用print函数：

```
print $number;
print $string;
```

如果希望将上下文中的变量值打印出来，可以将变量名放置在被打印的字符串中，只要使用双引号将它们正确引用即可（参见图2-5）：

```
print "Number is $number";
print "String is $string";
```

```
Number is 1
String is Hello, world!
```

图2-5　打印两个变量值的结果

print的这种用法只适用于标量（单值）变量类型，即数值和字符串型变量，而数组和对象是比较复杂的变量类型，不能仅使用print函数（参见图2-6）：

```
print "_SERVER is $_SERVER";
```

```
_SERVER is Array
```

图2-6　用print语句数组类的复杂变量类型，不能得到你想要的结果

你可以看到，print_r()可以处理这些非标量类型。在本章的后面，你还会学到其他处理方法。无论是处理标量变量还是非标量变量，在脚本出现问题时，打印出它们的值总是非常好的调试技术。

由于变量类型并不是锁定的（PHP被称为*弱类型语言*），因此它们可以在运行的时候被修改：

```
$variable = 1;
$variable = "Greetings";
```

如果现在打印$variable的值，结果将是Greetings。下面的脚本更好地诠释了为变量赋值并访问这些值的方法。

⇒ 为变量赋值并进行访问

(1) 在文本编辑器或IDE中创建新的PHP脚本，命名为variables.php（参看脚本2-3）。

脚本2-3　这个脚本定义了一些基本的变量，并且将它们的值打印出来

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Variables</title>
7   </head>
8   <body>
9   <?php // 脚本2-3 - variables.php
10
11  // 地址:
12  $street = "100 Main Street";
13  $city = "State College";
14  $state = "PA";
15  $zip = 16801;
16
17  // 打印地址:
18  print "<p>The address is:<br/>$street <br/>$city $state $zip</p>";
19
20  ?>
21  </body>
22  </html>
```

2

(2) 创建初始的HTML标签。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Variables</title>
</head>
<body>
```

(3) 开始编写PHP代码：

```
<?php // 脚本2-3 - variables.php
```

(4) 定义一些数值和字符串变量：

```
$street = "100 Main Street";
$city = "State College";
$state = "PA";
$zip = 16801;
```

这些代码创建了一些不同的字符串和数值类型的变量。字符串型使用引号来进行定义，并且每个变量名都遵循了正确的命名规则。

谨记，每个语句都要使用分号作为结尾，变量名是区分大小写的。

(5) 打印变量：

```
print "<p>The address is:<br/>$street <br/>$city $state $zip</p>";
```

这里使用一个`print`语句访问所有的变量。完整的需打印的字符串（由文本、HTML标签和

变量组成）被双引号所引用。HTML代码中的
标签使文本在浏览器中以多行的形式显示（请注意，在换行标签中额外的空白和斜线是为了符合XHTML的要求）。

(6) 完成PHP脚本部分和HTML页面：

```
?>
</body>
</html>
```

(7) 将文件保存为variables.php，并上传到服务器上（或者保存在本机合适的目录下），然后在Web浏览器上进行测试（参见图2-7）。

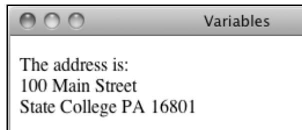


图2-7　图为一些被赋值的变量，在打印时带有上下文

✓提示

- 如果在运行此脚本时看到*语法解析*错误（参见图2-8），很可能是因为遗漏了分号，或者是有未配对的引号。

Parse error: syntax error, unexpected T_VARIABLE in **/Users/larryullman/Sites/phpvqs4/variables.php** on line **15**

图2-8　如你所见，语法解析错误是PHP中最常见的错误类型，它们通常由缺少分号和未配对的引号或圆括号引起

- 如果一个变量的值没有被打印出来，或者看到Undefined variable错误提示（参见图2-9），很可能是因为两次拼写变量名不一致所造成的拼写错误。
- 如果看到空白页，很可能是出了某些错误，同时PHP中display_errors的设置被设为off。参看第3章中的相关内容。

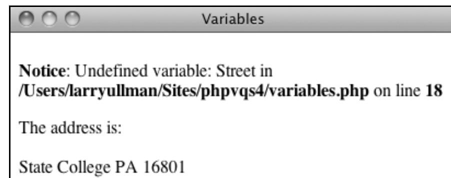


图2-9　Undefined variable错误意味着使用了一个没有值的变量（变量未被定义），这可能由拼写错误和大小写不一致引起

2.5　理解引号

既然已经对变量的基础知识以及如何创建变量有所了解，本书将开始阐述引号的重要概念。如大多数编程语言一样，PHP允许使用单引号（'）和双引号（"），但是它们将导致截然不同的

结果。理解这一点是非常重要的，因此在接下来的示例中将演示使用这两种引号的区别。

规则描述如下：使用单引号引用的内容将照字面意思进行处理，而被双引号引用的内容需要进行推断。也就是说，双引号引用的变量名将被它的值所替代，正如在脚本2-3中看到的那样。但单引号引用的变量名不会被替代。

这条规则在PHP里用到引号的情况中普遍适用，包括创建字符串变量和使用print函数。下面是解释这条规则的最佳示例。

⇒ 使用引号

(1) 在文本编辑器或者IDE中创建一段新的PHP脚本，命名为quotes.php（参看脚本2-4）。

脚本2-4 这段脚本清楚地诠释了使用不同类型的引号将导致的不同结果

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>Quotes</title>
7   </head>
8   <body>
9   <?php // 脚本2-4 - quotes.php
10
11  // 这里既可使用单引号，又可使用双引号:
12  $first_name = 'Larry';
13  $last_name = "Ullman";
14
15  // 这里单引号和双引号的作用不同:
16  $name1 = '$first_name $last_name';
17  $name2 = "$first_name $last_name";
18
19  // 这里单引号和双引号的作用不同:
20  print "<h1>Double Quotes</h1><p>name1 is $name1 <br/>
21  name2 is $name2</p>";
22
23  print '<h1>Single Quotes</h1><p>name1 is $name1 <br/>
24  name2 is $name2</p>';
25
26  ?>
27  </body>
28  </html>
```

(2) 创建初始HTML标签：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Quotes</title>
</head>
```

```
<body>
```

(3) 开始编写PHP代码：

```
<?php // 脚本2-4 - quotes.php
```

(4) 创建两个字符串变量：

```
$first_name = 'Larry';
$last_name = "Ullman";
```

为这两种变量使用单引号或者双引号没有任何区别，因为对于字符串都会取其字面值。但是，如果在这里使用自己的名字（完全可以）并且你的名字里包含一个撇号，那么要么需要使用双引号引用，要么将撇号转义并用单引号引用：

```
$last_name = "O'Toole";
$last_name = 'O\'Toole';
```

(5) 用`$first-name`和`$last-name`变量创建`$name1`和`$name2`变量：

```
$name1 = '$first_name $last_name';
$name2 = "$first_name $last_name";
```

在这里，使用不同的引号类型就会导致不同的结果。`$name1`变量会按字面处理，结果为`$first_name $last_name`，因为这里没有进行推断。相反，`$name2`的结果将是`Larry Ullman`，也就是预想中的结果。

(6) 使用两种引号类型打印变量：

```
print "<h1>Double Quotes</h1> <p>name1 is $name1 <br/>
name2 is $name2</p>";
print '<h1>Single Quotes</h1> <p>name1 is $name1 <br/>
name2 is $name2</p>';
```

不同类型的引号再次产生了不同的结果。第一个`print`语句打印出`$name1`和`$name2`变量的值，而第二个`print`语句则直接打印出`$name1`和`$name2`。

在`print`语句中的HTML代码让它们在浏览器中更易于阅读，并且每个语句都分两行打印变量的值，这让人更容易接受。

(7) 完成PHP代码段和HTML页面：

```
?>
</body>
</html>
```

(8) 将文件保存为`quotes.php`并上传至服务器（或者保存在本机合适的目录下），然后在Web浏览器上进行测试（参见图2-10）。

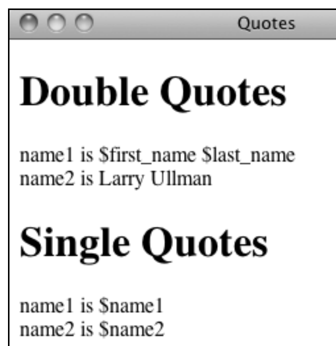


图2-10　不同的引号（单引号和双引号）指明需要打印的是变量的名称还是变量的值

✓提示

- 如果对于两种引号的区别仍然感到迷惑，坚持使用双引号将是个保险的做法。

- 可以认为使用单引号更为可取，因为PHP不需要通过搜索字符串来查找变量。这个规则更多的只是一种技巧——对性能影响非常小，可以忽略不计。
- 创建新行（`\n`）、回车（`\r`）和制表符（`\t`）的快捷方式都必须在双引号内使用以达到预期的效果。
- 请记住，并不是总需要使用引号的。当所赋的值为数值类型，或者只需要打印变量时，可以省略它们：

```
$num = 2;
print $num;
```


2.6 回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

2.6.1 回顾

- `$_SERVER`是什么类型的变量？
- 变量名是以什么打头的？
- 变量名的第一个字符可以使用哪些字符？在第一个字符后面可以使用哪些字符？
- 变量名区分大小写吗？
- 什么样的变量是标量？举出一些不同的标量变量类型。举出一个非标量变量类型。
- 赋值操作符是什么？
- 检查变量时，本章介绍的调试技术是什么？
- 双引号和单引号在使用上有什么不同之处？

2.6.2 实践

- 创建另一个PHP脚本，定义一些变量并打印它们的值。尽量使用不同标量类型的变量。
- 创建一个PHP脚本，在HTML代码中打印一些变量的值。为了加大难度，可以使用PHP和变量创建链接标签或图片标签。

第 3 章

HTML表单和PHP

本章内容

- ❑ 创建简单的表单
- ❑ 选择表单的提交方法
- ❑ 使用PHP接收表单数据
- ❑ 显示错误
- ❑ 错误报告
- ❑ 向页面手动发送数据
- ❑ 回顾和实践

第2章简要地介绍了变量。读者会经常创建自己的变量，也会经常使用同HTML表单关联的变量。表单是当今Web网站的基本单位，能够实现注册和登录系统、搜索功能、在线购物等功能和特性。甚至在最基本的网站里都能找到HTML表单的逻辑。同时，利用PHP就能非常简单地获得和处理由HTML表单生成的数据。

出于以上方面的考虑，本章将介绍创建HTML表单以及如何利用PHP访问表单中数据的基础知识。同时，本章还会介绍实际PHP编程中几个关键性的概念，其中包括如何处理脚本中的错误。

3.1 创建简单的表单

先创建一个反馈页面以获取用户的称呼、名称、电子邮件地址、反馈和备注（参见图3-1），这个页面将用在本章的HTML表单示例中。这里需要针对上面的要求创建所需的字段。这些用来生成表单的代码都包含在开启和关闭的标签中。

```
<form>
form elements
</form>
```

Please complete this form to submit your feedback:

Name: Mr. [ ]

Email Address: [ ]

Response: This is... ○ excellent ○ okay ○ boring

Comments: [ ]

Send My Feedback

图3-1 在本章示例中使用的HTML表单

form标签指明代码中一个表单的开头和结尾。每个表单中的元素都必须处于这对标记中。开启form标记还包含一个action属性，它用来指明将表单数据提交至哪个页面。这是在创建表

单时需要着重考虑的问题。在本书中，action属性通常将指向PHP脚本：

```
<form action="somepage.php">
```

在创建下一个表单之前，请重新简短回顾一下XHTML。如第1章所述，XHTML的一些规则导致它与HTML有截然不同的语法。对初学者来说，代码需要全部使用小写字母，并且每个标签属性都必须加引号引用。而且，每个标签都必须关闭，那些没有关闭形式的标签（如input）必须要在末尾处添加空格和斜线。因此，在HTML中可以这样书写：

```
<INPUT TYPE=TEXT NAME=address SIZE=40>
```

但是XHTML中需要这样书写：

```
<input type="text" name="address" size="40" />
```

本书希望这个简短的解释能帮助读者消除在下面这段代码中由XHTML引起的疑惑。

HTML和XHTML中的每个表单元素的名字必须唯一。HTML和XHTML的命名规则是一致的，只能使用字母、数字和下划线（_）。总而言之，名字应当符合逻辑且描述性强。

⇛ 创建基本的HTML表单

(1) 在文本编辑器或IDE中创建一个新的文档，命名为feedback.html（参看脚本3-1）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Feedback Form</title>
</head>
<body>
<!-- 脚本3-1 - feedback.html -->
<div><p>Please complete this form to submit your feedback:</p>
```

脚本3-1　这个HTML页面中有一个带有几个不同类型输入方式的表单

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Feedback Form</title>
7   </head>
8   <body>
9   <!-- 脚本3-1 - feedback.html -->
10  <div><p>Please complete this form to submit your feedback:</p>
11
12  <form action="handle_form.php">
13
14    <p>Name: <select name="title">
15    <option value="Mr.">Mr.</option>
16    <option value="Mrs.">Mrs.</option>
17    <option value="Ms.">Ms.</option>
```

```
18    </select> <input type="text" name="name" size="20" /></p>
19
20    <p>Email Address: <input type="text" name="email" size="20" /></p>
21
22    <p>Response: This is...
23    <input type="radio" name="response" value="excellent" /> excellent
24    <input type="radio" name="response" value="okay" /> okay
25    <input type="radio" name="response" value="boring" /> boring</p>
26
27    <p>Comments: <textarea name="comments" rows="3" cols="30"></textarea></p>
28
29    <input type="submit" name="submit" value="Send My Feedback" />
30
31  </form>
32  </div>
33  </body>
34  </html>
```

(2) 添加表单的开启标签：

```
<form action="handle_form.php">
```

这个表单标签指明将把表单提交至handle_form.php页面，它同这个HTML页面在相同的目录下。如果希望更加明晰，可以在PHP脚本中使用完整的URL路径（例如，http://www.example.com/handle_form.php）。

(3) 为填写人名添加一个选择菜单以及一个文本输入框：

```
<p>Name: <select name="title">
<option value="Mr.">Mr.</option>
<option value="Mrs.">Mrs.</option>
<option value="Ms.">Ms.</option>
</select> <input type="text"
name="name" size="20" /></p>
```

这个用以输入人名的部分将由两个元素组成（参见图3-1）。第一个元素是一个选项为常用称谓的下拉菜单，选项为：Mr.、Mrs.和Ms。每个列举在select标签中的选项都可供用户进行选择（参见图3-2）。第二个元素是一个基本文本框，用来填写完整名称。

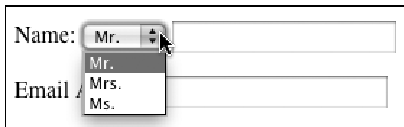


图3-2　该选择元素创建了一个带有选项的下拉菜单

(4) 添加一个文本框用来填写用户的电子邮件地址：

```
<p>Email Address: <input type="text" name="email" size="20" /></p>
```

(5) 为反馈添加单选按钮：

```
<p>Response: This is...
```

```
<input type="radio" name="response" value="excellent" /> excellent
<input type="radio" name="response" value="okay" /> okay
<input type="radio" name="response" value="boring" /> boring</p>
```

这段HTML代码创建了3个单选按钮（可以点击的圆圈，参见图3-1）。由于它们的name值相同，因此一次只能选择3个中的1个。在XHTML规则中，除了值以外代码都用小写字母形式，并且需要在每个input标签的末尾处使用一个额外的空格和斜线以关闭该标签。

(6) 添加一个textarea用来记录备注：

```
<p>Comments: <textarea name="comments" rows="3" cols="30"></textarea></p>
```

textarea能够比文本框提供更多的空间以供用户填写备注等文字。但是文本框可以对用户能够输入的信息数量进行限制，这是textarea所不能做到的（确切地说是在没有使用JavaScript的情况下）。当创建表单时，需要依照从用户获取信息的不同方式选择相应的input控制类型。

请注意，textarea没有关闭标签。

(7) 添加提交按钮：

```
<input type="submit" name="submit" value="Send My Feedback" />
```

提交元素的value属性将显示在Web浏览器中相应的按钮上（参见图3-1）。也可以使用例如Go!或Enter的字样。

(8) 关闭表单：

```
</form>
```

(9) 完成页面：

```
</div>
</body>
</html>
```

(10) 将页面保存为feedback.html并在浏览器上查看。

由于这是个HTML页面而不是PHP脚本，因此可以在自己的计算机上用Web浏览器直接查看。

✓提示

- ❑ 请注意，在这里使用HTML扩展名（.html）是因为它是一个标准的HTML页面（而不是PHP页面）。也可以使用.php扩展名，这不会造成任何问题，即便是页面里并不包含任何PHP代码。（请记住，PHP标签外面的代码——<?php和?>——都将被视作HTML。）
- ❑ 确认action属性正确地指向在服务器上实际存在的文件，否则表单将不能被正常提交。在这种情况下，需要指明应该把该表单提交至handle_form.php，它同feedback.html页位于相同的目录下。
- ❑ 在这个示例中，一个HTML表单被手动编码的HTML创建，如果愿意的话，还可以在Web页面应用程序中创建表单（例如，Adobe Dreamweaver）。
- ❑ 在即将到来的HTML 5中，增添了一些非常受欢迎的新表单元素，如Email、url和number。

3.2　选择表单的 method

经验丰富的HTML开发人员将会注意到，这个反馈表单缺少一件东西：初始表单标记中没有method属性。这个属性是用来告诉服务器如何将数据从表单传送至处理脚本的。

method有两种方法可供选择：GET和POST。许多HTML程序员并不完全清楚它们之间的区别，以及何时该用哪个。GET和POST之间的区别在于将信息从表单向处理信息的脚本传递的方式。GET将所有的信息聚集起来并作为URL的一部分进行传递，而POST并不让用户看到传递的信息。例如，当提交表单时，如果使用GET方式，URL看上去就可能像这样：

http://www.example.com/page.php? some_var=some_value&age=20&...

而使用POST方式时，最终用户将只能看到：

http://www.example.com/page.php

当在两种方式中进行选择时，请考虑下面的4个因素：

- 使用GET方式时，传送的信息量有限；
- GET方式公开地将输入的信息向处理脚本传送（这意味着，诸如密码这样的信息都将会被任何注视Web浏览器的人看到，这将引发严重的安全问题）；
- 使用GET方式的表单所创建的页面能够添加为书签，但是POST方式却不可以；
- 如果用户试图重载通过POST访问的页面，他就会收到提示信息（参见图3-3）；但重载通过GET访问的页面就不会收到提示信息。



图3-3　如果用户刷新PHP脚本，且该脚本的数据是通过POST方式发送的，他就会收到系统的确认信息（具体的信息取决于使用的浏览器）

一般来说，从服务器上请求信息时，使用GET请求。几乎所有用于搜索的页面都会使用GET（当你使用搜索引擎时，看一下它的URL），就像那些分页显示结果的页面一样（比如分类浏览产品）。相反，POST一般用于触发基于服务器的行为，比如提交一张联系表（发送Email）或提交某个博客的评论（评论添加到数据库和该博客的页面上）。

尽管GET方法也有一些有用的技术（参看3.6节），但是本书将专门使用POST方法来处理表单。

⇒ 向脚本添加方法

(1) 在文本编辑器或者IDE中打开feedback.html文件（参看脚本3-1）。

(2) 在初始表单标签中添加method="post"（参看脚本3-2中第12行）。

脚本3-2　添加值为POST的method属性以完成表单

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3    <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4    <head>
5      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6      <title>Feedback Form</title>
7    </head>
8    <body>
9    <!-- 脚本3-2 - feedback.html -->
10   <div><p>Please complete this form to submit your feedback:</p>
11
12   <form action="handle_form.php" method="post">
13
14     <p>Name: <select name="title">
15     <option value="Mr.">Mr.</option>
16     <option value="Mrs.">Mrs.</option>
17     <option value="Ms.">Ms.</option>
18     </select> <input type="text" name="name" size="20" /></p>
19
20     <p>Email Address: <input type="text" name="email" size="20" /></p>
21
22     <p>Response: This is...
23     <input type="radio" name="response" value="excellent" /> excellent
24     <input type="radio" name="response" value="okay" /> okay
25     <input type="radio" name="response" value="boring" /> boring</p>
26
27     <p>Comments: <textarea name="comments" rows="3" cols="30"></textarea></p>
28
29     <input type="submit" name="submit" value="Send My Feedback" />
30
31   </form>
32   </div>
33   </body>
34   </html>
```

表单的method属性告诉浏览器如何将表单数据发送到接收脚本。因为在提交表单时包含很多数据（包括备注），并且因为用户并不会将结果页面添加为书签，所以在这里使用POST方法更为合理。

(3) 保存脚本，并在Web浏览器上重新加载。

应该养成在修改了脚本之后，立即在Web浏览器中重新加载页面的习惯。重新加载页面这一步常常会被忽略，导致你对更改后的页面没有任何变化而迷惑不解。

(4) 查看页面的源代码以确定所有的元素都出现在页面上并且具有正确的属性（参见图3-4）。

✓提示

- 在讨论方法的时候，GET和POST都使用了大写字母以引起注意。但是，在脚本中的表单内，要使用POST以保持同XHTML的一致性。不要为这种不一致感到担心（如果发现存在这样的情况），method属性并不区分大小写。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <title>Feedback Form</title>
</head>
<body>
<!-- Script 3.2 - feedback.html -->
<div><p>Please complete this form to submit your feedback:</p>

<form action="handle_form.php" method="post">

        <p>Name: <select name="title">
        <option value="Mr.">Mr.</option>
        <option value="Mrs.">Mrs.</option>
        <option value="Ms.">Ms.</option>
        </select> <input type="text" name="name" size="20" /></p>

        <p>Email Address: <input type="text" name="email" size="20" /></p>

        <p>Response: This is...
        <input type="radio" name="response" value="excellent" /> excellent
        <input type="radio" name="response" value="okay" /> okay
        <input type="radio" name="response" value="boring" /> boring</p>

        <p>Comments: <textarea name="comments" rows="3" cols="30"></textarea></p>

        <input type="submit" name="submit" value="Send My Feedback" />

</form>
</div>
</body>
</html>
```

图3-4　在表单中，很多重要的信息（如action和method的值或者元素的名称）都只能在HTML源代码中看到

3.3　使用 PHP 接收表单数据

前两节我们创建了一个基本的HTML表单，现在需要编写PHP脚本，它将接收和处理表单数据。在这个示例中，PHP脚本将重复用户在表单中输入的内容。第4章将介绍如何获得这些信息并将它们保存到数据库中，如何通过电子邮件发送这些信息以及如何将信息写入文件，等等。

访问被提交的表单数据，需要查阅第2章中关于预定义变量的内容，在那里已经介绍了预定义变量$_SERVER。这个PHP脚本用来处理表单数据而引用的特定变量可能是$_GET或$_POST中的一个。如果HTML表单使用GET方法，那么将能够通过$_GET找到提交的表单数据。如果HTML表单使用POST方法，那么被提交的表单数据将存储在$_POST中。

$_GET和$_POST除了是预定义变量（也就是不需要创建）外，还是数组和一个特殊的变量类型。这就意味着每个变量都可能包含大量的值。不能这样使用数组（参见图2-6）：

```
print $_POST; // 无法工作!
```

相反，为了访问一个特殊的值，必须引用数组的索引或者键。在第7章将对此进行详细介绍，但是这个前提真的非常简单。从一个name属性的值为something的表单元素开始：

```
<input type="text" name="something" />
```

现在，假设表单使用POST方法，在表单元素中输入的值在$_POST['something']中将是

有效的：

```
print $_POST['something'];
```

遗憾的是，这里存在一个小问题：当在双引号中使用单引号引用键时，将引发解析错误（参见图3-5）。

```
print "Thanks for saying:$_POST['something']";
```

Parse error: syntax error, unexpected T_ENCAPSED_AND_WHITESPACE, expecting T_STRING or T_VARIABLE or T_NUM_STRING in **/Users/larryullman/Sites/phpvqs4/handle_form.php** on line **19**

图3-5　这个难看的解析错误是由试图在双引号中使用$_POST['something']引起的

有几种方法可以避免这样的问题。在本章中，将选用语法最为简单的一种方法：只是先把这个特殊的$_POST元素赋值给另一个变量：

```
$something = $_POST['something'];
print "Thanks for saying: $something";
```

在新的PHP脚本中执行这个信息前最后还有两点需要注意的是。首先，$_POST是区分大小写的：它必须严格地像你所看到的这样输入：一个美元符号、一个下划线，然后是所有大写的字母。其次，在先前的示例中，$_POST的索引——必须严格地同提供给表单元素的name值匹配。

⇛ 处理HTML表单

(1) 在文本编辑器或IDE中创建一个新的文档，命名为handle_form.php（参看脚本3-3）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Your Feedback</title>
</head>
<body>
```

脚本3-3　此脚本显示通过引用关联的$_POST变量向脚本提交的表单数据

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8"/>
6     <title>Your Feedback</title>
7   </head>
8   <body>
9   <?php // 脚本3-3 handle_form.php
10
11  // 该页面从feedback.html接收数据。
```

```
12  // 将会接收：标题、姓名、Email、反映、评论并用$_POST提交。
13
14  // 为新的变量赋值：
15  $title = $_POST['title'];
16  $name = $_POST['name'];
17  $response = $_POST['response'];
18  $comments = $_POST['comments'];
19
20  // 打印接收到的数据：
21  print "<p>Thank you, $title $name, for your comments.</p>
22  <p>You stated that you found this example to be '$response' and
    added:<br/>$comments</p>";
23
24  ?>
25  </body>
26  </html>
```

(2) 添加PHP开启标签和一些注释：

```
<?php // 脚本3-3 handle_form.php
// 该页面从feedback.html接收数据。
// 将会接收：标题、姓名、Email、反映、评论并用$_POST提交。
```

向脚本添加注释是为了让它的执行目的更加清晰。在feedback.html页面中指明了数据被发送的目的地(通过action属性),同时注释从相反的方向进行阐述(脚本从哪里接收这些数据)。

(3) 用接收到的数据为新的变量赋值：

```
$title = $_POST['title'];
$name = $_POST['name'];
$response = $_POST['response'];
$comments = $_POST['comments'];
```

当表单使用POST方法时，提交的数据能够在$_POST数组中找到。可使用语法$_POST['name_attribute_value']访问这个独立的值。这些操作将不会受到表单元素类型（文本框、下拉框、复选框等）的影响。

为了能够更加容易地在第(4)步print语句中使用这些值，在这里每个变量都将被赋予新的值。本书没有对$_POST['email']或$_POST['submit']做任何处理，如果需要的话，你可以加上一些处理。

(4) 将用户信息打印出来：

```
print "<p>Thank you, $title $name, for your comments.</p>
<p>You stated that you found this example to be '$response' and added:
→<br/>$comments</p>";
```

这个print语句在上下文中使用了4个变量，用来向用户显示脚本接收了什么数据。

(5) 关闭PHP代码片段并且完成HTML页面：

```
?>
</body>
</html>
```

(6) 将脚本保存为handle_form.php。

注意文件名必须与`action`属性的值完全相同。

(7) 将脚本上传至服务器（如果在自己的计算机上安装了PHP，也可以保存在本机适当的目录下），确保保存在同`feedback.html`相同的目录下。

(8) 通过URL（http://*something*）加载`feedback.html`。

必须通过URL来加载HTML表单，这样当向PHP脚本提交表单时，PHP脚本也能够通过URL运行。注意：PHP脚本必须通过URL运行！

忘记通过URL加载表单是初学者常常会犯的错误。

(9) 填写表单（参见图3-6），完成后提交（参见图3-7）。

Please complete this form to submit your feedback:

Name: Mr. Larry Ullman

Email Address: larry@example.com

Response: This is... excellent okay boring

Comments: No problems so far!

Send My Feedback

图3-6　用户在HTML表单中输入的所有内容都应该通过`handle_form.php`脚本显示在Web浏览器中（参见图3-7）

Thank you, Mr. Larry Ullman, for your comments.

You stated that you found this example to be 'excellent' and added:
No problems so far!

图3-7　这是在第1章中讨论过的`print`语句的另一种应用，它创建了第一个动态生成的网页

如果看到空白页，阅读3.4节，了解如何显示可能发生的错误。

如果看到一个错误通知（参见图3-8）或者看到一个变量在打印输出的时候没有值，很有可能是因为表单元素的`name`值或者`$_POST`数组的索引拼写错误（或者没有将表单填写完整）。

Notice: Undefined index: Name in **/Users/larryullman/Sites/phpvqs4/handle_form.php** on line **16**

Thank you, Mr. , for your comments.

图3-8　如果以某种形式使用了并不存在的变量，那么就会出现如图所示的通知。在本例中，发生错误是因为将`$_POST['name']`写成`$_POST['Name']`了

✓提示

- 如果希望向PHP脚本传递预设值，可以在HTML表单中使用文本输入框的隐藏类型。例如，在表单标签中插入代码行：

```
<input type="hidden" name="form_page"
value="feedback.html" />
```

将创建一个值为feedback.html的变量，它在处理脚本的时候将调用$_POST['this_page']。

- 请注意，单选钮和选择菜单变量的值是基于被选择项的value属性（例如，单选钮的excellent值），这对于复选框也适用。对于文本框，变量的值就是用户输入的内容。
- 如果handle_form.php脚本在提交的字符串中显示出多余的斜线，请查看“Magic Quote”框注中提供的解释和解决方案。

无论表单使用什么method，你都可以在预定义的$_REQUEST变量中访问表单数据。然而，$_GET和$_POST更明确，所以更可取。

Magic Quote

在PHP的早期版本中有名为Magic Quote的特性，而这一特性已经过时了（你不应该再使用这一特性了，它迟早会被删除）。当开启Magic Quote时，它会对提交的表单数据中的单引号和双引号自动进行转义处理。因此字符串I’d like more information将会被转换成为I\’d like more information。

对存在潜在问题的字符进行转义非常有用，甚至在某些情况下显得非常必要。但是如果在安装PHP时开启Magic Quote特性，将在PHP脚本打印出表单数据时看到这些反斜杠。可以使用stripslashes()函数来撤销这种效果，将其应用在handle_form.php脚本中，可以这样做：

```
$comments = stripslashes($_POST ['comments']);
```

用来代替：

```
$comments = $_POST['comments'];
```

这将对转义后被提交的字符串产生消除转义的效果，使之回到没有转义之前的原始状态。如果提交的表单数据中没有出现附加的斜线，就不必去管Magic Quote。

3.4　显示错误

在调试PHP脚本时出现的首要问题是，在这里不一定能看到发生的错误。在Web服务器上安装PHP之后，它将在默认的安全配置下运行，处理数据的方式、表现的性能等都将在这个环境中进行。默认设置中的一项是不显示发生的任何错误。换句话说，display_errors设置处于关闭状态（参见图3-9）。在这种情况下，如果脚本发生错误，看到的将是一个空白页（这是全新安装的PHP的标准设置，大多数托管的公司都会把display_errors设置为开启状态）。

不在活动的网站里显示错误的原因是为了规避一些安全风险。简单地说，PHP的错误显示通常会给公众提供过于详尽的信息。但是，作为开发人员来说，必须能够查看这些错误以修正它们。

display_errors	Off	Off
display_startup_errors	On	On
doc_root	*no value*	*no value*
docref_ext	*no value*	*no value*
docref_root	*no value*	*no value*
enable_dl	On	On
error_append_string	*no value*	*no value*
error_log	/Applications/MAMP/logs /php_error.log	/Applications/MAMP/logs /php_error.log
error_prepend_string	*no value*	*no value*
error_reporting	32767	32767

图3-9　运行phpinfo()脚本（如脚本1-2），查看在服务器中display_errors的设置状态

为了能够显示PHP错误，可以进行以下操作：

- 开启display_errors设置（参看附录A以获取更多信息）；
- 为某些单独的脚本开启display_errors设置。

当开发一个网站时，到目前为止第一个选项是首选。但是，这个选项仅仅是针对拥有服务器管理员权限的用户。任何人都可以通过在脚本中包含以下代码行来选用第二个选项：

```
ini_set ('display_errors', 1);
```

ini_set()函数允许脚本临时覆盖PHP配置文件中的设置。在这个示例中，将开启display_errors设置，在这里用数字1代表开启状态。

尽管任何人都可以使用第二种方式，但是其不足之处在于：如果脚本发生某些特定类型的错误（将在第4章中进行讨论），将不会被执行。因此，这行代码也就不会被执行，并且这种特定错误，以及任何阻止脚本运行的因素都将导致出现一个空白页面。

⇛ 显示错误的步骤

(1) 在文本编辑器或IDE中打开handle_form.php。

(2) 在PHP代码的第一行，输入以下代码（参看脚本3-4）：

```
ini_set ('display_errors', 1);
```

脚本3-4　向PHP脚本添加的这个代码段将开启display_errors设置，它将使发生的所有错误都被显示出来

```
1    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3    <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4    <head>
5      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6      <title>Your Feedback</title>
7    </head>
8    <body>
9    <?php // 脚本3-4 - handle_form.php #2
10
```

```
11  ini_set ('display_errors', 1);
    // 让我从错误中学到知识!
12
13  // 该页面从feedback.html接收数据。
14  // 将会接收：标题、姓名、Email、反映、评论并用$_POST提交。
15
16  // 为新的变量赋值：
17  $title = $_POST['title'];
18  $name = $_POST['name'];
19  $response = $_POST['response'];
20  $comments = $_POST['comments'];
21
22  // 打印接收到的数据：
23  print "<p>Thank you, $title $name, for
    your comments.</p>
24  <p>You stated that you found this example to be '$response' and added:
    <br/>$comments</p>";
25
26  ?>
27  </body>
28  </html>
```

该行代码将告诉PHP希望看到发生的任何错误。需要在PHP代码段首先对它进行调用，以便剩下的PHP代码都遵从这个新的设定。

(3) 将文件保存为`handle_form.php`。

(4) 上传文件至Web服务器，并在Web浏览器上进行测试（参见图3-10和图3-11）。

Please complete this form to submit your feedback:

Name: Mr. Larry Ullman

Email Address:

Response: This is... excellent okay boring

Comments:

Send My Feedback

图3-10　再次尝试打开这个表单

Notice: Undefined index: response in **/Users/larryullman/Sites/phpvqs4/handle_form.php** on line **19**

Thank you, Mr. Larry Ullman, for your comments.

You stated that you found this example to be '' and added:

图3-11　现在发生的任何错误都被显示出来。生成的通知表明引用的表单元素没有值

如果结果页中没有任何错误，脚本将像先前那样运行。如果之前运行表单时看到了空白页，现在将可能看到图3-11中显示的消息。如果看到这样的错误，很可能是因为表单元素的名称拼写错误、`$_POST`数组的索引拼写错误，或者是没有完整地填写表单。

✓提示

- 确保在进行任何困难的脚本调试时，display_errors都处于开启状态。如果在计算机上安装了PHP，本书强烈建议在学习时开启其中的display_errors设置（请再次参考附录A）。
- 如果运行PHP脚本时看到了空白页，也需要检查HTML源代码以查找错误或其他问题。
- ini_set()函数只能用来更改某些特定的设置。参看PHP手册以了解更多细节。
- 请记住，display_errors只控制是否要向Web浏览器发送错误消息。它并不产生错误也不会阻止错误的发生。

如果在name表单元素后面漏掉等号，也会出现问题：

```
<input name"something" />
```


3.5 错误报告

需要了解与display_errors有关的另一个PHP配置问题是**错误报告**。在PHP中有11种不同的错误，而在版本6中，另外还有4种用户定义的类型（本书将不做介绍）。表3-1中列举了最重要的4种常见错误类型，并对它们进行了描述和举例。

表3-1 PHP错误类型

类　型	描　述	示　例
通知	非致命性错误，是或者不是表明有问题存在	引用一个没有值的变量
警告	非致命性错误，通常表明有问题存在	函数误用
解析错误	由语法错误导致的致命性错误	缺少分号，或者引号、圆括号和花括号
错误	一般性的致命错误	内存分配问题

有两种方法设置PHP报告哪些错误。第一种方法是在PHP的配置文件中调整error_reporting()级别（参见附录A）。如果你使用的是自己的PHP服务器，在开发脚本时可能需要调整服务器的全局设置。

第二种方法是在脚本中使用error_reporting()函数，该函数带有一个数字或者一些常量（未使用引号引用的字符串，具有预先指定的意义）来调整级别。表3-2列出了比较重要的一些常量。

表3-2 错误报告常量

名　称	名　称
E_NOTICE	E_ERROR
E_WARNING	E_ALL
E_PARSE	E_STRICT

使用这些信息，可以向脚本添加下面的代码：

```
error_reporting (0);
```

```
error_reporting (E_ALL);
error_reporting (E_ALL & ~E_NOTICE);
```

第一行代码指明不需要报告任何错误。第二行代码要求报告所有的错误。最后一个示例说明需要看到除notice之外的所有错误消息。请牢记，调整这些设置不会阻止或者产生错误，它只是影响是否报告错误。

通常使用最高级的错误报告对于开发和测试PHP脚本来说是最好的选择。可以声明希望看到所有的错误以及strict错误报告：

```
error_reporting (E_ALL | E_STRICT);
```

E_ALL设置并不包含E_STRICT，这就是为什么这行代码表明所有的错误信息将被显示，或者（竖线，称为*管道符*）strict错误被显示的原因。后面的设置对错误提供了更进一步的报告，但是也可能会成为在PHP未来版本中引发notice提示问题的原因。让我们在handle_form.php页面中应用这个设置。

⇒ 调整错误报告

(1) 在文本编辑器或IDE中打开handle_form.php（参看脚本3-4）。

(2) 在ini_set()的下1行，添加下面的代码（参看脚本3-5）：

```
error_reporting (E_ALL | E_STRICT);
```

脚本3-5　调整脚本的错误报告级别可以对潜在的和已经存在的问题提供或多或少的反馈

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8"/>
6     <title>Your Feedback</title>
7   </head>
8   <body>
9   <?php // 脚本3-5 - handle_form.php #3
10
11  ini_set ('display_errors', 1);
    // 让我从错误中学到知识!
12  error_reporting (E_ALL | E_STRICT);
    // 显示所有可能出现的问题!
13
14  // 该页面从feedback.html接收数据。
15  // 将会接收：标题、姓名、Email、反映、评论并用$_POST提交。
16
17  // 为新的变量赋值。
18  $title = $_POST['title'];
19  $name = $_POST['name'];
20  $response = $_POST['response'];
21  $comments = $_POST['comments'];
22
```

```
23  // 打印接收到的数据:
24  print "<p>Thank you, $title $name, for your comments.</p>
25  <p>You stated that you found this example to be '$response' and added:
    <br/>$comments</p>";
26
27  ?>
28  </body>
29  </html>
```

(3) 将文件保存为handle_form.php。

(4) 在启用了PHP的服务器上将文件保存至适当的目录，并通过提交表单在Web浏览器上进行测试（参见图3-12和图3-13）。

Please complete this form to submit your feedback:

Name: Ms. Blankenship

Email Address: ida@example.edu

Response: This is... excellent okay boring

Comments: Enough already!

Send My Feedback

图3-12 再次测试表单

Thank you, Ms. Blankenship, for your comments.

You stated that you found this example to be 'boring' and added:
Enough already!

图3-13 试验的结果（如果填写完整，并且没有任何编程错误）

此时此刻，如果表单填写完整，并且$_POST的索引同表单元素的name值精确匹配，将看不到任何错误发生（正如图中显示的那样）。如果有问题存在，包含任何潜在的问题（由于有E_STRICT），它们都应该显示出来并产生错误报告。

✓提示

- PHP手册列举了所有的错误报告级别，但是在这里列举出来的都是其中最重要的错误类型。
- 本书中的代码都是使用最高级别的错误报告（E_ALL|E_STRICT）测试的。

3.6 向页面手动发送数据

本章的最后一个示例和其他的主题比起来有些跑题，但就如何使用PHP处理表单数据提供了额外的补充。正如在3.2节中讨论的那样，如果表单使用GET方法，URL将会像这样：

http://www.example.com/page.php? some_var=some_value&age=20&...

向接收页面（在这里是page.php）发送了一系列诸如name=value的信息对，它们中的每一个都被与号（&）分隔开。整个序列由一个问号开始（紧随在正在处理的脚本名称之后）。

为了用这种方式访问传递给页面的值，可以使用$_GET变量。正如使用$_POST那样，引用这个特定的名称作为$_GET的索引。在这个示例中，page.php接收了值为some_value的$_GET['some_ var']变量，值为20的$_GET['age']变量，等等。

正如本书介绍的那样，可以通过创建一个使用GET方法的HTML表单来传递数据。使用这种方法，也可以向PHP页面发送数据，而无需使用表单。通常情况下，可以通过在另一页中使用链接的方法做到：

```
<a href="page.php?id=22">Some Link</a>
```

在从数据库中读取数据后，这个链接能够由PHP动态生成，它将向page.php传递值22并可访问$_GET['id']。

下面两段代码将通过使用一个硬编码的HTML页面诠释这个概念，请自行进行尝试。

⇛ 创建HTML页面

(1) 在文本编辑器或IDE中创建一个新的文档，命名为hello.html（参看脚本3-6）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Greetings!</title>
</head>
<body>
<!-- 脚本3-6 - hello.html -->
<div><p>Click a link to say hello:</p>
```

脚本3-6　这个HTML页面使用链接来向PHP脚本传递值（由此效仿使用GET方法的表单）

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Greetings!</title>
7   </head>
8   <body>
9   <!-- 脚本3-6 - hello.html -->
10  <div><p>Click a link to say hello:</p>
11
12  <ul>
13    <li><a href="hello.php?name=Michael">Michael</a></li>
14    <li><a href="hello.php?name=Celia">Celia</a></li>
15    <li><a href="hello.php?name=Jude">Jude</a></li>
16    <li><a href="hello.php?name=Sophie">Sophie</a></li>
```

```
17  </ul>
18
19  </div>
20  </body>
21  </html>
```

(2) 创建一个链接指向PHP脚本，通过下面的URL传递值：

```
<ul>
  <li><a href="hello.php?name=Michael">Michael</a></li>
  <li><a href="hello.php?name=Celia">Celia</a></li>
  <li><a href="hello.php?name=Jude">Jude</a></li>
  <li><a href="hello.php?name=Sophie">Sophie</a></li>
</ul>
```


这样做的前提是用户看到一个链接的列表，每个链接都同一个特定的名称关联（参见图3-14）。当用户点击链接时，将会在URL中把与名称对应的值发送至`hello.php`（参见图3-15）。

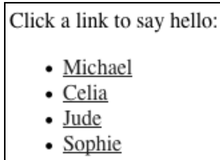


图3-14　这个简单的HTML页面拥有4个指向PHP脚本的链接

```
<!-- Script 3.6 - hello.html -->
<div><p>Click a link to say hello:</p>

<ul>
        <li><a href="hello.php?name=Michael">Michael</a></li>
        <li><a href="hello.php?name=Celia">Celia</a></li>
        <li><a href="hello.php?name=Jude">Jude</a></li>
        <li><a href="hello.php?name=Sophie">Sophie</a></li>
</ul>

</div>
```

图3-15　HTML源代码显示出这些值是如何通过4个链接进行发送的

如果希望使用不同的名称，这样很好，但是请坚持使用只有一个单词的名称形式，而不要在中间加上空格或者标点符号（否则它们不能被正确地传送至PHP脚本，我们将对此进行说明）。

(3) 完成HTML页面：

```
</div>
</body>
</html>
```

(4) 将脚本保存为`hello.html`并且在启用了PHP的服务器上保存至适当的目录。

(5) 在Web浏览器中通过URL加载这个HTML页面。

可以通过点击页面上的链接访问PHP脚本，这样就能不用输入URL而查看到HTML页面，但是在这里请从使用URL开始。先不要点击任何链接，因为PHP脚本并不存在。

⇒ 创建PHP脚本

(1) 在文本编辑器或IDE中创建一个新的文档，命名为hello.php（参看脚本3-7）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Greetings!</title>
</head>
<body>
```

脚本3-7　这个PHP页面在URL中引用了name的值，用来打印一个问候

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Greetings!</title>
7   </head>
8   <body>
9   <?php // 脚本3-7 - hello.php
10
11  ini_set ('display_errors', 1); // 让我从错误中学到知识！
12  error_reporting (E_ALL | E_STRICT); // 显示所有可能出现的问题！
13
14  // 该页面将从URL中接收name的值。
15
16  // 打印Hello:
17  $name = $_GET['name'];
18  print "<p>Hello, <span style=\"font-weight: bold;\">$name </span>!</p>";
19
20  ?>
21  </body>
22  </html>
```

(2) 开始编写PHP代码：

```
<?php // 脚本3-7 - hello.php
```

(3) 如果需要，对错误管理进行说明：

```
ini_set ('display_errors', 1);
error_reporting (E_ALL | E_STRICT);
```

这两行代码对PHP如何进行错误响应做了设置，这在本节之前已经做过介绍了。它们不是必须的，但是会有所帮助。

(4) 在URL中传递name的值来创建一个问候：

```
$name = $_GET['name'];
print "<p>Hello, <span style= \"font-weight: bold;\">$name </span>!</p>";
```

通过URL将name变量传送给页面（参看脚本3-6）。引用$_GET['name']以访问这个值。可以使用$_GET（同$_POST相对），这是因为值是通过GET请求传送的。

如同之前的PHP脚本一样，预定义变量（$_GET）中的值被首先赋给另外一个变量，以简化在print语句中的语法。

(5) 完成PHP代码和HTML页面：

```
?>
</body>
</html>
```

(6) 将脚本保存为hello.php并且在启用了PHP的服务器上保存到适当的目录。它应该与hello.html（脚本3-6）保存在相同的目录下。

(7) 点击hello.html中的链接，查看结果（参见图3-16和图3-17）。

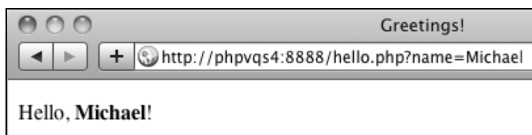


图3-16　通过点击第一个链接，Michael作为name的值在URL中进行传送，最终出现在问候中



图3-17　通过点击最后一个链接，Celia作为name的值在URL中进行传送，最终出现在问候中

✓提示

- 如果直接运行hello.php，将获得一个错误通知，这是因为没有name的值将通过URL进行传送（参见图3-18）。



图3-18　如果$_GET['name']变量没有被赋值，浏览器将打印出这样带有错误通知的难堪信息

- 由于hello.php通过URL读取值，因此它独立于hello.html运行。例如，可以直接编辑hello.php中的URL来使用name的值进行问候，即使hello.html并不包含指向那样name的链接（参见图3-19）。

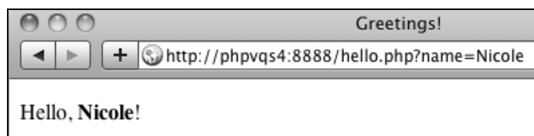


图3-19　在URL中赋给name（小写）的任何值，都会被PHP脚本问候

- 如果希望使用链接向脚本传送多个值，将variable=value对（例如，first_name = Larry）用与号（&）进行分隔。因此这里的链接将是hello.php? first_name=Larry& last_name=Ullman。在学习urlencode()函数（后面会介绍）之前，最好继续使用单个单词，不要添加标点或空格。
- 虽然这里的示例为个人的name设置了值，这也许并不实际，但是这种基础技术在很多场景中却非常有用。例如，一个PHP脚本可以构成一个模板，根据页面从URL中接收到的值的不同，产生具有不同内容的Web页面。

3.7　回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

3.7.1　回顾

- 表单中action属性的作用是什么？
- 表单中method属性的作用是什么？GET与POST哪个更安全？哪个方法可以在浏览器中添加为书签。
- 哪些预定义变量包括从表单提交的数据？注意：答案不止一个。
- 某个HTML页面包括提交到PHP脚本的表单，为什么必须通过URL加载该HTML页面。
- 在哪种情况下，在脚本中尝试启用display_errors会失败？为什么在实际网站上启用display_errors会不安全？

3.7.2　实践

- 不通过URL在Web浏览器中加载feedback.html（如在地址栏中以file://开头填地址），填写表单后提交。仔细观察结果。你应该能够识别问题，并了解问题的原因，这样今后出现类似问题时，你就可以解决了。
- 确保在你的开发环境中启用display_errors。
- 确保在你的开发环境中error_reporting已经设置为E_ALL | E_STRICT。
- 尝试在一个PHP脚本中引入不同的问题（例如，引号不对称、漏掉分号、错误地引用变量等），再观察结果。

- 做一个实验，向hello.html和hello.php页面传入不同的值（包括数字），通过URL提交到PHP脚本。
- 在hello.html上创建一个变量，发送多个name=value对到PHP脚本。PHP脚本是否打印所有接收到的值？
- 如果你想知道能够传入的值类型，又不介意等待的话，可以试着通过URL向页面传入一些复杂的值（如空格或标点），看看会发生什么。
- 创建一个新的HTML表单，用它执行一个假想的任务（或是一个更简单的任务）。创建PHP脚本处理该表单，打印收到的数据。

第 4 章

使用数值

4

本章内容

- ❑ 创建表单
- ❑ 执行算术运算
- ❑ 格式化数值
- ❑ 理解优先级
- ❑ 数值的自增和自减
- ❑ 创建随机数
- ❑ 回顾和实践

在第2章中，我们讨论了变量的不同类型，如何为它们赋值以及在通常情况下变量的用法。我们将专门在本章介绍两种数值型变量——整型（整数）和浮点数值（又称为浮点或小数）。

本章首先讲述创建HTML表单，它用来创建一些数值型变量。然后将开始学习如何执行基本算术操作、如何对数值进行格式化，以及如何处理运算符优先级。本章的最后两节将涉及自增和自减数值以及生成随机数。本章通篇都将讨论与数值相关的PHP实用函数。

4.1　创建表单

本章中大多数PHP示例都将在电子商务前提下执行各种计算。一个表单可能会接受价格、数量、折扣值、税率以及运输费（参见图4-1），并且处理表单的PHP脚本将会返回总体费用。这个费用将会以用户希望分期付款的期数进行拆解，从而生成月费用的值（参见图4-2）。

Fill out this form to calculate the total cost:

Price: 5.00

Quantity: 100

Discount: 10.00

Tax: 7.5 (%)

Shipping method: Slow and steady

Number of payments to make: 10

Calculate!

图4-1　这个表单接受从用户获取的数值，并把它们发送给PHP页面

You have selected to purchase:
100 widget(s) at
$5.00 price each plus a
$5.00 shipping cost and a
7.5 percent tax rate.
After your $**10.00** discount, the total cost is $**532.13**.
Divided over **10** monthly payments, that would be $**53.21** each.

图4-2 PHP脚本会根据用户提交的数据计算费用并返回结果。这是本章结束时此表单的运算结果

让我们从创建一个HTML页面开始学习的历程，用户可以通过这个页面输入不同的值。

⇛ 创建HTML表单的步骤

(1) 在文本编辑器或IDE中创建一个新的HTML文档，命名为calculator.html（参看脚本4-1）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Product Cost Calculator </title>
</head>
<body><!-- 脚本4-1 - calculator.html -->
<div><p>Fill out this form to calculate the total cost:</p>
```

脚本4-1 这个基本的HTML表单创建了将在PHP脚本中执行算术计算的数值

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Product Cost Calculator</title>
7   </head>
8   <body><!-- 脚本4-1 - calculator.html -->
9   <div><p>Fill out this form to calculate the total cost:</p>
10
11  <form action="handle_calc.php" method="post">
12
13  <p>Price: <input type="text" name="price" size="5" /></p>
14
15  <p>Quantity: <input type="text" name="quantity" size="5" /></p>
16
17  <p>Discount: <input type="text" name="discount" size="5" /></p>
18
19  <p>Tax: <input type="text" name="tax" size="3" /> (%)</p>
20
21  <p>Shipping method: <select name="shipping">
22  <option value="5.00">Slow and steady </option>
23  <option value="8.95">Put a move on it. </option>
24  <option value="19.36">I need it yesterday!</option>
```

```
25  </select></p>
26
27  <p>Number of payments to make: <input type="text" name="payments" size="3" /></p>
28
29  <input type="submit" name="submit" value="Calculate!" />
30
31  </form>
32
33  </div>
34  </body>
35  </html>
```

(2) 创建初始form标签：

```
<form action="handle_calc.php" method="post">
```

这个form标签标志着HTML表单的开始。它的action属性指明表单数据将被提交给名为handle_calc.php的页面。该标签的method属性告诉页面使用POST方式来传送数据。参看第3章以获取更多的有关信息。

(3) 为price、quantity、discount和tax创建文本框：

```
<p>Price: <input type="text" name="price" size="5" /></p>
<p>Quantity: <input type="text" name="quantity" size="5" /></p>
<p>Discount: <input type="text" name="discount" size="5" /></p>
<p>Tax: <input type="text" name="tax" size="3" /> (%)</p>
```

HTML没有给数值提供输入类型，因此需要创建文本框以供值的输入。插入的说明指明tax的格式是百分数。

请记住，文本输入框使用的name命名方式应当同有效的PHP变量名命名方式相符（只能是字母、数字和下划线，并且不能以数字开头，等等)。

(4) 添加一个字段供用户选择shipping的方式：

```
<p>Shipping method: <select name="shipping">
<option value="5.00">Slow and steady</option>
<option value="8.95">Put a move on it.</option>
<option value="19.36">I need it yesterday!</option>
</select></p>
```

这个shipping选项是用下拉菜单实现的。每个选项的值就是选择该选项所对应的shipping方式产生的费用。因此，如果用户选择诸如Put a move on it选项时，$_POST['shipping']在handle_calc.php中获取到的值将是8.95。

(5) 完成HTML表单：

```
<p>Number of payments to make: <input type="text" name="payments"
→size="3" /></p>
<input type="submit" name="submit" value="Calculate!" />
</form>
```

最后两个输入区域接受了分期付款所需的期数，并且创建了一个提交按钮（标签为Calculate!)。关闭form标签标志着页面中表单部分的结束。

(6) 完成HTML页面。

```
</div>
</body>
</html>
```

(7) 将脚本保存为`calculator.html`并在Web浏览器上查看。

由于这是一个HTML页面，因此可以在Web浏览器上直接查看。

4.2　算术运算

正如在小学里学到的那样，基础数学涉及加、减、乘、除的法则。这些运算法则都在PHP中使用最显而易见的运算符表示：

- 加 (+)；
- 减 (–)；
- 乘 (*)；
- 除 (/)。

演示上述运算符的使用方法，我们将创建一个PHP脚本来计算一些小商品售卖的总费用。用以计算的脚本将是一个基本的购物车应用程序——一种非常实用的Web页面特性（但在这个示例中，相关数值变量的值将来自于`calculator.html`）。

编写这个脚本时，请务必注意注释的用法（参看脚本4-2），它在代码后阐明了不同代码行的意义以及生成的原因。

脚本4-2　PHP脚本用表单提交的数值进行了所有的标准算术计算

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Product Cost Calculator</title>
7     <style type="text/css" media="screen">
8       .number { font-weight: bold; }
9     </style>
10  </head>
11  <body>
12  <?php // 脚本4-2 - handle_calc.php
13  /* 脚本calculator.html获取数值
14  并计算总成本和月付款数。*/
15
16  // 可以在此进行错误处理。
17
18  // 从$_POST数组获取数值：
19  $price = $_POST['price'];
20  $quantity = $_POST['quantity'];
21  $discount = $_POST['discount'];
22  $tax = $_POST['tax'];
23  $shipping = $_POST['shipping'];
```

```
24  $payments = $_POST['payments'];
25
26  // 计算总额:
27  $total = $price * $quantity;
28  $total = $total + $shipping;
29  $total = $total - $discount;
30
31  // 定义税率:
32  $taxrate = $tax/100;
33  $taxrate = $taxrate + 1;
34
35  // 纳入税率后的总成本:
36  $total = $total * $taxrate;
37
38  // 计算月付款数:
39  $monthly = $total / $payments;
40
41  // 打印结果:
42  print "<p>You have selected to purchase:<br/>
43  <span class=\"number\">$quantity </span> widget(s) at <br/>
44  $<span class=\"number\">$price</span>
    price each plus a <br/>
45  $<span class=\"number\">$shipping </span> shipping cost and a <br/>
46  <span class=\"number\">$tax</span>  percent tax rate.<br/>
47  After your $<span class=\"number\"> $discount</span> discount, the total
    cost is
48  $<span class=\"number\">$total </span>.<br/>
49  Divided over <span class=\"number\"> $payments</span> monthly payments,
    that would be $<span class=\"number\"> $monthly</span> each.</p>";
50
51  ?>
52  </body>
53  </html>
```

⇒创建销售费用计算器

(1) 在文本编辑器或IDE中创建一个新的文档，命名为handle_calc.php（参看脚本4-2）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Product Cost Calculator </title>
  <style type="text/css"  media="screen">
    .number { font-weight: bold;}
  </style>
</head>
<body>
```

在脚本的开头处定义了一个名为number的CSS类。页面中任何带有这个类值的元素都将被加粗。换句话说，当脚本打印输出时，表单中的输入数值和各种计算结果都会以更明显的方式（粗体）显示出来。

(2) 如果需要，可以插入PHP标签和地址错误处理：

```
<?php // 脚本4-2 - handle_calc.php
```

可以依照当前PHP的设置情况，添加一些代码行来开启display_errors，并调整错误报告的级别。参看第3章中的相关信息。

(3) 将$_POST元素的值赋值给局部变量：

```
$price = $_POST['price'];
$quantity = $_POST['quantity'];
$discount = $_POST['discount'];
$tax = $_POST['tax'];
$shipping = $_POST['shipping'];
$payments = $_POST['payments'];
```


脚本将获取到在预定义的$_POST变量中所有的表单数据。为了访问单个的表单值，引用$_POST['index']以将index替换为相应表单元素的name值。这些值在这里被赋给单个的局部变量，以便在脚本剩余部分更加方便地使用它们。

请注意，每个变量都被给予了一个描述性名称，并且全部使用小写字母命名。

(4) 开始计算总费用：

```
$total = $price * $quantity;
$total = $total + $shipping;
$total = $total - $discount;
```

星号（*）在PHP中代表乘法，因此total首先被计算为购买的数量（$quantity）乘以单价（$price）。运输费（$shipping）需要添加进总费用中（请记住，运输费与shipping下拉菜单中所选择项的value属性相关），并且减去折扣（$discount）。

注意在为变量赋值的时候，可以在其中使用变量的现有值，这是完全可以接受的（正如在最后两行中所做的那样）。

(5) 计算税率和新的总费用：

```
$taxrate = $tax/100;
$taxrate = $taxrate + 1;
$total = $total * $taxrate;
```

税率（$tax）应当是一个百分比，例如将8或者5.75除以100，转化为同百分比等同的小数（0.08或者0.0575）。最后将这个结果加上1计算购买物品的实际税率，然后乘以先前计算的总价。这在算术上相当于用小数类型的税率（$taxrate）乘以总费用（$total），得到的结果成为总费用（$total）的新值（例如，消费100美元需缴纳5%的税，因此总共将花费105美元，这就相当于用100美元直接乘以1.05得到的结果）。

(6) 计算月付费用：

```
$monthly = $total / $payments;
```

作为除法的示例，我们假设购买某物或者任何需要分期付款的购买行为将会持续数月。因此，我们需要将总费用除以需要分期付款的月数以获得月付费用。

(7) 打印结果：

```
print "<p>You have selected to purchase:<br/>
<span class=\"number\">$quantity </span> widget(s) at <br/>
$<span class=\"number\">$price </span> price each plus a <br/>
$<span class=\"number\">$shipping </span> shipping cost and a <br/>
<span class=\"number\">$tax</span> percent tax rate.<br/>
After your $<span class=\"number\"> $discount</span> discount, the total cost is
$<span class=\"number\">$total </span>.<br/>
Divided over <span class=\"number\"> $payments</span> monthly payments,
→that would be $<span class=\"number\">$monthly</span> each.</p>";
```

`print`语句将每个值连同一些文本一起发送到Web浏览器。为了增加可读性，增加`<br/>`标签以格式化浏览器的显示结果。另外，`print`函数将结果处理为多行显示，以让PHP代码看上去更加清晰。每个变量的值在浏览器中都被突出显示，使用的方法是用带有`number`类属性的`Span`标记对它们进行折行。

(8) 关闭PHP代码段并且完成HTML页面。

```
?>
</body>
</html>
```

(9) 将脚本保存为`handle_calc.php`，并放在启用了PHP的服务器上适当的目录下。

确保同`calculator.html`在同一目录下。

(10) 在Web浏览器中测试脚本（参见图4-3和图4-4）。

本书不想在这点上做过多冗长的解释，但是请确保你确实是通过URL（http://*something*）来加载HTML表单的，这样当它被提交时，PHP脚本可以通过这个URL运行。

可以用这些值进行试验，验证计算器是否正常运行。如果漏掉了一些值，结果将显得有一些怪异，但是计算仍将继续进行（参见图4-5）。

Fill out this form to calculate the total cost:

Price: 19.95

Quantity: 6

Discount: 10.00

Tax: 6 (%)

Shipping method: Slow and steady

Number of payments to make: 12

Calculate!

图4-3　HTML表单如图显示

You have selected to purchase:
6 widget(s) at
$**19.95** price each plus a
$**5.00** shipping cost and a
6 percent tax rate.
After your $**10.00** discount, the total cost is $**121.582**.
Divided over **12** monthly payments, that would be $**10.1318333333** each.

图4-4　计算结果

You have selected to purchase:
6 widget(s) at
$**19.95** price each plus a
$**5.00** shipping cost and a
percent tax rate.
After your $ discount, the total cost is $**124.7**.
Divided over **12** monthly payments, that would be $**10.3916666667** each.

图4-5　可以漏掉或者更改任何值然后重新运行计算器。这里将漏掉`tax`和`discount`的值

✓提示

- 你一定会注意到，计算器计算出来的结果同真实的美元值并不相符（参见图4-4和图4-5）。4.3节将介绍如何处理这样的结果。
- 如果希望在税值或者折扣值(或两者都)前打印总费用的值,可以使用两种方法。在$total变量的值被再次变更之前，在合适的值后插入适当的print语句。或者使用新变量存储后来计算结果的值（例如，$total_with_tax和$total_less_discount）。
- 由于变量以美元符号开头，因此在打印带有美元符号的数字时（比如$10，10通过变量产生）需要格外小心。不能使用$$variable变量，这是因为两个美元符号的组合将产生一个非常复杂的变量类型，而本书将不对其进行讨论。解决方案是在美元符号中间放置一些东西。例如，在本示例中，美元符号和变量名之间被放置了一个空格或者HTML标签。另一个选择是将第一个美元符号进行转义：

  ```
  print "The total is \$$total";
  ```

  第三种选择是使用连接字符，这会在第5章介绍。
- 这个脚本会根据提交字段的不同显示不同的结果。唯一可能产生问题的字段是月付的数量：如果它被漏掉，将会看到除以零的警告。第6章将讨论在使用表单数据前对它们的有效性进行验证。
- HTML 5中预计会有一个或多个限制用户输入数值的文本框。

4.3　格式化数值

尽管已经可以使用计算器,但是现在仍然存在一个合理性的问题：月付金额不能是10.13183333美元的形式。为了生成更多可用的数值，需要对它们进行格式化。

有两个函数适合达到格式化数值的目的。第一个是round()，它用于对数值截取特定位数的小数。函数的第一个参数是需要格式化的数值，它既可以是一个数字，也可以是一个存有数值的变量。第二个参数是可选的，它代表需要取的小数位数。例如：

```
round (4.30); // 4
round (4.289, 2); // 4.29
$num = 236.26985;
round ($num); // 236
```

另外一个可以用来格式化数值的函数是number_format()。number_format()同round()的工作方式类似，它有一个数值参数（或者有数字值的变量）以及一个可选的小数位数指定参数。这个函数通过千位分组来格式化数字，格式化的效果通常如下所示：

```
number_format (428.4959, 2); // 428.50
number_format (428, 2); // 428.00
number_format (123456789); // 123,456,789
```

让我们将数值进行适当的格式化并重新编写PHP脚本。

⇒ 格式化数值的步骤

(1) 在文本编辑器或IDE中打开handle_calc.php（参看脚本4-2）。

(2) 在所有计算代码之后、print语句之前，添加下面的代码（参看脚本4-3）：

```
$total = number_format ($total, 2);
$monthly = number_format ($monthly, 2);
```

脚本4-3　对数值变量的值应用number_format()函数，以让它们符合实际使用的要求

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Product Cost Calculator</title>
7     <style type="text/css" media="screen">
8       .number { font-weight: bold;}
9     </style>
10  </head>
11  <body>
12  <?php // 脚本4-3 - handle_calc.php #2
13  /* 脚本从calculator.html获取数值
14  并计算总成本和月付款数。*/
15
16  // 可以在此进行错误处理。
17
18  // 从$_POST数组获取数值：
19  $price = $_POST['price'];
20  $quantity = $_POST['quantity'];
21  $discount = $_POST['discount'];
22  $tax = $_POST['tax'];
23  $shipping = $_POST['shipping'];
24  $payments = $_POST['payments'];
25
26  // 计算总额：
27  $total = $price * $quantity;
28  $total = $total + $shipping;
29  $total = $total - $discount;
30
31  // 定义税率：
32  $taxrate = $tax/100;
33  $taxrate = $taxrate + 1;
34
35  // 纳入税率后的总成本：
36  $total = $total * $taxrate;
37
38  // 计算月付款数：
39  $monthly = $total / $payments;
40
41  // 格式化数值，保留2位小数。
42  $total = number_format ($total, 2);
43  $monthly = number_format ($monthly, 2);
```

```
44
45  // 打印结果：
46  print "<p>You have selected to purchase:<br/>
47  <span class=\"number\">$quantity</span> widget(s) at <br/>
48  $<span class=\"number\">$price</span> price each plus a <br/>
49  $<span class=\"number\">$shipping</span> shipping cost and a <br/>
50  <span class=\"number\">$tax</span> percent tax rate.<br/>
51  After your $<span class=\"number\"> $discount</span> discount, the total cost is
52  $<span class=\"number\">$total </span>.<br/>
53  Divided over <span class=\"number\"> $payments</span> monthly payments, that
    would be $<span class=\"number\">$monthly </span> each.</p>";
54
55  ?>
56  </body>
57  </html>
```


格式化这些数值，可以在每个计算操作完成之后应用这些函数，但是一定要在它们被传送给Web浏览器之前。第二个参数（2）指明结果数值需要保留2位小数，这个设置会对数值四舍五入，并且会在需要的时候用零补位。

(3) 将文件保存在同calculator.html相同的目录下，并在浏览器上测试（参见图4-6和图4-7）。

Fill out this form to calculate the total cost:

Price: 99.00

Quantity: 4

Discount: 25.00

Tax: 5.5 (%)

Shipping method: Put a move on it.

Number of payments to make: 24

Calculate!

图4-6 另外一次执行表单的结果

You have selected to purchase:
4 widget(s) at
$99.00 price each plus a
$8.95 shipping cost and a
5.5 percent tax rate.
After your **$25.00** discount, the total cost is **$400.85**.
Divided over **24** monthly payments, that would be **$16.70** each.

图4-7 脚本的更新版本返回了更多符合实际的值，这得益于number_format()函数处理的结果

✓提示

- 此外，还有很多更加复杂的方式都使用printf()和sprintf()函数来格式化数值。由于它们的语法比较复杂，因此本书将不进行讨论。请参看PHP手册以获得更多的相关信息。
- 非Windows版本的PHP有money_format()函数，它可以用在有number_format()场合。
- 出于一些复杂的原因，round()函数对于“精确”半数的情况（0.5、0.05、0.005等），舍去和进上的次数是一半对一半[①]。

① 这里的意思是如果需要round()函数处理的数字其小数部分正好是一半，则取整到最接近的偶数，并遵守“四舍六入五成双”原则，即前一位是奇数，则进1，前一位是偶数则舍入。因此，Round(1.5)=2；Round(2.5)=2；Round(0.15)=0.2；Round(0.005)=0。——译者注

- 在PHP中调用函数时，函数名和封装参数的圆括号之间可以有（也可以没有）空格，两种写法都正确：

```
round ($num);
round($num);
```

- `number_format()`函数接受两个可选参数，这两个参数分别用来指定使用什么字符代表小数点位数和千分号。这是非常有用的，例如，在某些文化的习惯中，将1 000.89写为1.000 89。如果希望使用这些选项，请参看PHP手册中相关的语法描述。

4.4　理解优先级

在讨论不同类型的算术运算符时，必定会涉及优先级的问题。优先级是指一系列计算的执行顺序。例如，下面变量的值是多少?

```
$number = 10 - 4 / 2;
```

`$number`的值是3（10减去4等于6，然后被2除）还是8（4被2除后等于2，10再减去2等于8）？这里的结果是8，因为除法的优先级要高于减法。

附录B中提供了PHP里运算符优先级的完整列表（包括这里尚未介绍到的运算符）。然而，可以使用圆括号对来绕过所有的概念和规则，而不是去尝试记住那个有特殊字符的庞大表格。圆括号对的优先级通常要高于所有其他运算符。因此：

```
$number = (10 - 4) / 2; // 3
$number = 10 - (4 / 2); // 8
```

在算式中使用圆括号能够确保不会看到因为优先级问题引发的特殊（错误）结果，也可以使用圆括号来将复杂的算式重写为更少行数的代码。让我们使用圆括号重写`handle_calc.php`脚本，将多行代码合并为一行，同时确保精度。

⇒ 管理优先级

(1) 在文本编辑器或者IDE中打开`handle_calc.php`（参看脚本4-3）。

(2) 改变总费用第一次计算的方式（参看脚本4-4）：

```
$total = (($price * $quantity) + $shipping) - $discount;
```

脚本4-4　使用圆括号能够改写压缩多行算式（参看脚本4-3），而不会影响脚本的算术精度

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Product Cost Calculator</title>
7     <style type="text/css" media="screen">
8       .number { font-weight: bold;}
9     </style>
10  </head>
```

```
11  <body>
12  <?php // 脚本4-4 - handle_calc.php #3
13  /* 脚本从calculator.html获取数值
14  并计算总成本和月付款数。*/
15
16  // 可以在此进行错误处理。
17
18  // 从$_POST数组获取数值:
19  $price = $_POST['price'];
20  $quantity = $_POST['quantity'];
21  $discount = $_POST['discount'];
22  $tax = $_POST['tax'];
23  $shipping = $_POST['shipping'];
24  $payments = $_POST['payments'];
25
26  // 计算总额:
27  $total = (($price * $quantity) + $shipping) - $discount;
28
29  // 定义税率:
30  $taxrate = ($tax/100) + 1;
31
32  // 纳入税率后的总成本:
33  $total = $total * $taxrate;
34
35  // 计算月付款数:
36  $monthly = $total / $payments;
37
38  // 格式化数值，打印2位小数。
39  $total = number_format ($total, 2);
40  $monthly = number_format ($monthly, 2);
41
42  // 打印结果:
43  print "<p>You have selected to purchase:<br/>
44  <span class=\"number\">$quantity</span> widget(s) at <br/>
45  $<span class=\"number\">$price</span> price each plus a <br/>
46  $<span class=\"number\">$shipping</span> shipping cost and a <br/>
47  <span class=\"number\">$tax</span> percent tax rate.<br/>
48  After your $<span class=\"number\"> $discount</span> discount, the total cost is
49  $<span class=\"number\">$total </span>.<br/>
50  Divided over <span class=\"number\"> $payments</span> monthly payments, that
    would be $<span class=\"number\">$monthly </span> each.</p>";
51
52  ?>
53  </body>
54  </html>
```

只要使用圆括号确保算术运算能够正确执行，就不用逃避将所有的运算放在一个步骤中。另外一个选择是记住PHP中多个运算符的优先级规则，但是使用圆括号还是要简便一些。

(3) 改变税值的计算方式：

```
$taxrate = ($tax/100) + 1;
```

这里将原有的2行税值计算代码合并为1行。

(4) 在同calculator.html相同的目录下保存脚本，并且在Web浏览器上测试（参见图4-8和图4-9）。

Fill out this form to calculate the total cost:

Price: 1.50

Quantity: 250

Discount: 0

Tax: 6 (%)

Shipping method: I need it yesterday!

Number of payments to make: 2

Calculate!

图4-8 再次测试表单

You have selected to purchase:
250 widget(s) at
$1.50 price each plus a
$19.36 shipping cost and a
6 percent tax rate.
After your **$0** discount, the total cost is **$418.02**.
Divided over **2** monthly payments, that would be **$209.01** each.

图4-9 尽管运算被精简，但是计算结果却没有发生改变。如果看到不同的结果或者得到一个错误消息，就要再次检查圆括号的匹配情况（打开和关闭的圆括号数量应当一致）

✓提示

- ❑ 在创建算式时，要确保圆括号对始终都是匹配的（每个打开的圆括号都需要有一个关闭的圆括号）。否则将会出现解析错误。
- ❑ 倘若使用在这里应用的这个方法，可以将所有的总费用计算合并为只有1行代码（而不是3行），但是这样会让一些事情过于简单化。

4.5 数值的自增和自减

如同Perl和其他大多数编程语言一样，PHP包含很多快捷方式，用来避免产生一些不美观的结构，例如：

```
$tax = $tax + 1;
```

当某个变量的值需要增加1（被成为增量调整）或者减少1（被成为减量调整）时，可以使用++或者--，下面分别给出二者的示例：

```
$var = 20; // 20
$var++; // 21
$var++; // 22
$var--; // 21
```

这里再一次重写handle_calc.php脚本，专门测试这个概念。

⇒ 变量值的自增

(1) 在文本编辑器或者IDE中打开handle_calc.php（参看脚本4-4）。

(2) 改变脚本4-3中税率的计算方法，更改结果如下所示（参看脚本4-5）：

```
$taxrate = $tax/100;
$taxrate++;
```

脚本4-5 数值的自增和自减是一种常见的运算方式，它们分别用++或者--表示

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Product Cost Calculator</title>
7     <style type="text/css" media="screen">
8       .number { font-weight: bold;}
9     </style>
10  </head>
11  <body>
12  <?php // 脚本4-3 - handle_calc.php #4
13  /* 该脚本从calculator.html获取数值
14  并计算总成本和月付款数。*/
15  
16  // 可以在此进行错误处理。
17  
18  // 从$_POST数组获取数值:
19  $price = $_POST['price'];
20  $quantity = $_POST['quantity'];
21  $discount = $_POST['discount'];
22  $tax = $_POST['tax'];
23  $shipping = $_POST['shipping'];
24  $payments = $_POST['payments'];
25  
26  // 计算总额:
27  $total = (($price * $quantity) + $shipping) - $discount;
28  
29  // 定义税率:
30  $taxrate = $tax/100;
31  $taxrate++;
32  
33  // 纳入税率后的总成本:
34  $total = $total * $taxrate;
35  
36  // 计算月付款数:
37  $monthly = $total / $payments;
38  
39  // 格式化数值，保留2位小数。
40  $total = number_format ($total, 2);
41  $monthly = number_format ($monthly, 2);
42  
43  // 打印结果:
44  print "<p>You have selected to purchase:<br/>
45  <span class=\"number\">$quantity</span> widget(s) at <br/>
46  $<span class=\"number\">$price</span> price each plus a <br/>
47  $<span class=\"number\">$shipping</span> shipping cost and a <br/>
48  <span class=\"number\">$tax</span> percent tax rate.<br/>
49  After your $<span class=\"number\"> $discount</span> discount, the total cost is
```

4

```
50  $<span class=\"number\">$total </span>.<br/>
51  Divided over <span class=\"number\"> $payments</span> monthly payments, that
    would be $<span class=\"number\">$monthly </span> each.</p>";
52
53  ?>
54  </body>
55  </html>
```

第一行通过将`$tax`的值除以100计算出了税率。第二行将该结果加`1`，使其与总值相乘后能够得到带税的总值。

(3) 在同`calculator.html`相同的目录下保存脚本，并在Web浏览器中进行测试（参见图4-10和图4-11）。

Fill out this form to calculate the total cost:

Price: 5.00

Quantity: 100

Discount: 10.00

Tax: 7.5 (%)

Shipping method: Slow and steady

Number of payments to make: 10

Calculate!

图4-10 最后一次执行表单

You have selected to purchase:
100 widget(s) at
$5.00 price each plus a
$5.00 shipping cost and a
7.5 percent tax rate.
After your $**10.00** discount, the total cost is $**532.13**.
Divided over **10** monthly payments, that would be $**53.21** each.

图4-11 使用长的或者短的变量自增版本，将不会对计算结果产生影响（比较脚本4-4和脚本4-5）

✓提示

- 虽然从功能上看，编写代码既可以使用`$taxrate=$taxrate+1;`方法，也可以使用精简的`$taxrate++`，但后一种方式（使用自增操作符）更加专业并且通用。
- 在第6章中，将看到自增操作符如何被普遍地同循环联合使用。

算术赋值运算符

PHP也提供一些算术运算符和赋值运算符的组合，它们是`+=`、`-=`、`*=`和`/=`。每一个都通过执行运算来为变量赋值。例如以下两行代码都为变量值加5：

```
$num = $num + 5;
$num += 5;
```

也就是说，`handle_calc.php`脚本可以用下述语句决定税率：

```
$tax = $_POST['tax']; // 5
$tax /= 100; // 现在$tax是0.05
$tax += 1; // 1.05
```

这种简略的运算赋值语句你会经常见到。

4.6 创建随机数

本章介绍的最后一个函数是rand()，它是一个随机数生成工具，用于输出随机数：

```
$n = rand(); // 31
$n = rand(); // 87
```

如果需要将产生的数值限定在一个特定范围内，rand()函数可以通过接受最小值和最大值的参数做到：

```
$n = rand (0, 10);
```

参数的值在包含范围内，因此在上面的例子中0和10都可能是返回的值。

让我们创建一个简单的脚本“Lucky Numbers”作为随机数生成的示例。

⇛ 创建随机数的步骤

(1) 在文本编辑器或IDE中创建一个新文档，命名为random.php（参看脚本4-6）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Lucky Numbers</title>
</head>
<body>
```

脚本4-6 rand()函数生成随机数

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Lucky Numbers</title>
7   </head>
8   <body>
9   <?php // 脚本4-6 - random.php
10  /* 这个脚本生成3个随机数。*/
11
12  // 可以在此进行错误处理。
13
14  // 创建3个随机数:
15  $n1 = rand (1, 99);
16  $n2 = rand (1, 99);
17  $n3 = rand (1, 99);
18
19  // 打印随机数:
20  print "<p>Your lucky numbers are:<br/>
21  $n1<br/>
22  $n2<br/>
23  $n3</p>";
```

```
24
25  ?>
26  </body>
27  </html>
```

(2) 如果需要，如下添加PHP标签和错误管理：

```
<?php // 脚本4-6 - random.php
```

(3) 创建3个随机数：

```
$n1 = rand (1, 99);
$n2 = rand (1, 99);
$n3 = rand (1, 99);
```

这些数字通过3次单独调用`rand()`函数产生，并且将每个结果赋值给了不同的变量。

(4) 打印这些数字：

```
print "<p>Your lucky numbers are:<br/>
$n1<br/>
$n2<br/>
$n3</p>";
```

这个`print`语句非常简单。数字被打印出来，通过在它们之前加上一个HTML的break标签每一个数字都在不同的行里显示。

(5) 关闭PHP代码和HTML页面：

```
?>
</body>
</html>
```

(6) 将脚本在启用了PHP的服务器上适当的目录中保存为`random.php`，然后在Web浏览器中进行测试（参见图4-12和图4-13）。

Your lucky numbers are:
32
68
71

图4-12　通过调用`rand()`函数产生了3个随机数

Your lucky numbers are:
23
81
2

图4-13　再次运行脚本，将获得不同的结果

✓提示

- `getrandmax()`函数将返回使用`rand()`可能产生的随机数中最大的数值。结果的值会因为所使用的操作系统的不同而有所不同。
- PHP有另外一个用以生成随机数的函数`mt_rand()`。它同`rand()`非常相似（但是更好），并且在对于生成密码这样的敏感场景来说是更好的选择。请查看PHP手册以查找关于`mt_rand()`函数的更多和更完整的主题讨论。

其他的算术函数

PHP拥有大量用以处理算术数据的内置函数。本章已经介绍了round()、number_format()和rand()。

PHP将round()分为两种函数。第一种是ceil()，将生成的每个随机数值向最接近的整数进行“入”运算。第二种是floor()，将生成的每个随机数值向最接近的整数进行“舍”运算。

计算页面可以使用的另外一个函数是abs()，它将返回数值的绝对值。在这种情况下不用记住绝对值，它就可以做到：

```
$number = abs(-23); // 23
$number = abs(23); // 23
```

用外行的话来说，数值的绝对值通常是正数。

除了这些函数，PHP还提供可能用到的所有的三角函数、指数函数、基数转换和对数函数。参看PHP手册以获得更多的信息。

4.7　回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

4.7.1　回顾

- 四个主要的算术运算符是什么？
- 下面的代码为什么不能工作：

  ```
  print "The total is $$total";
  ```

  如何修改？
- 如果HTML页面包括要提交到PHP脚本的表单，为什么必须通过URL加载该HTML？
- 哪些函数可以用来格式化数值？如何对数值截取特定位数的小数。
- 运算符优先级的重要性是什么？
- 自增、自减运算符是什么？
- 有哪些算术赋值运算符？

4.7.2　实践

- 在PHP手册中查找一个本章新学的函数。单击页面上的链接，浏览其他与数值有关的函数。
- 创建一个新的包含数值的HTML表单。然后创建PHP脚本接收表单数据，执行一些计算，格式化数值，并打印结果。

第 5 章

使用字符串

5

本章内容

- 创建HTML表单
- 连接字符串
- 处理换行符
- HTML和PHP
- 字符串的编码和解码
- 查找子字符串
- 替换局部字符串
- 回顾和实践

第2章已介绍过，PHP中使用的第二种变量类型是*字符串*，它是用单引号或者双引号括起来的一系列的字符。一个字符串变量可以包含一个单独的字母、一个单词、一个句子、一个段落、HTML代码，甚至是一些没有意义的字符、数字和符号（可能代表一串密码）。字符串也许是PHP中最常用的变量类型。

本章将涵盖PHP中最基础的内置函数和操作字符串数据的运算符，不论这些字符串是来自于表单还是由脚本最初声明的。本章还将介绍一些常见技术。例如，字符串修剪（如去掉空白和其他预定义字符）、字符串连接和字符串编码。字符串的其他用法将在后续的章节中诠释。

5.1　创建 HTML 表单

如同第3章，首先创建一个HTML表单，这个表单将向PHP脚本传送一些字符串变量的值。这里用以诠释理论的示例是一个在线的公告板或者论坛，用户可以在上面发布信息、他们的Email地址和姓名（参见图5-1）。

⇛ 创建一个HTML表单

(1) 在文本编辑器或者IDE中创建一个新的HTML文档，命名为`posting.html`（参看脚本5-1）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Forum Posting</title>
</head>
<body>
<!-- 脚本5-1 - posting.html -->
<div><p>Please complete this form to submit your posting:</p>
```

图5-1 这个HTML表单是本章大多数示例的基础

脚本5-1 这个表单向PHP脚本传送字符串数据

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Forum Posting</title>
7   </head>
8   <body>
9   <!-- 脚本5-1 - posting.html -->
10  <div><p>Please complete this form to submit your posting:</p>
11
12  <form action="handle_post.php" method="post">
13
14    <p>First Name: <input type="text" name="first_name" size="20" /></p>
15
16    <p>Last Name: <input type="text" name="last_name" size="20" /></p>
17
18    <p>Email Address: <input type="text" name="email" size="30" /></p>
19
20    <p>Posting: <textarea name="posting" rows="9" cols="30"></textarea></p>
21
22    <input type="submit" name="submit" value="Send My Posting" />
23
```

```
24  </form>
25  </div>
26  </body>
27  </html>
```

(2) 创建表单的初始标签：

```
<form action="handle_post.php" method="post">
```

这个表单将会将数据用POST的方式传送给handle_post.php脚本。

(3) 为First Name、Last Name和Email Address添加文本框标签：

```
<p>First Name: <input type="text" name="first_name" size="20" /></p>
<p>Last Name: <input type="text" name="last_name" size="20" /></p>
<p>Email Address: <input type="text" name="email" size="30" /></p>
```

这些都是第3章中介绍过的基本文本输入类型。请记住，不同文本框标签的name值应当符合PHP变量的命名规则（不能有空格、不能以数字开头、只能是字母、数字和下划线的任意组合）。

(4) 为posting添加一个文本框标签：

```
<p>Posting: <textarea name="posting" rows="9" cols="30"></textarea></p>
```

posting字段是一个textarea（文本区），它提供比文本输入框更大的输入空间。

(5) 创建提交按钮并关闭表单：

```
<input type="submit" name="submit" value="Send My Posting" />
</form>
```

每个表单都必须有一个提交按钮（或提交图片）。

(6) 完成HTML页面：

```
</div>
</body>
</html>
```

(7) 将文件保存为posting.html，放置在启用了PHP的服务器上的适当目录中，并在Web浏览器中进行查看（参见图5-1）。

这是一个HTML页面，因此它不必在启用了PHP的服务器上查看。但是由于它最终会向PHP脚本发送数据，因此最好还是将它保存在服务器上。

✓提示

- 从技术上说，除了上传的文件之外，所有表单数据都会被作为字符串发送给处理脚本。这包括在文本框中输入的数值数据、被选中的下拉菜单选项、复选框或者单选按钮的值，等等。例如，第4章的表单，将带有数值的字符串发送到脚本。
- 很多用PHP编写的论坛系统都是可以免费使用的。本书将不涉及如何进行完整的论坛开发，但是可以在我编写的《PHP 6与MySQL 5基础教程》中找到多语言论坛的开发方法。
- 本书的Web站点中有一个供读者发布问题和其他读者（以及作者）回答问题的论坛，可以在www.LarryUllman.com/forum/list.php?30找到。

5.2 连接字符串

连接（concatenation）是一个难以驾驭，但却是非常有用的概念。它用来将不同的项串联到一起。在编程时，特指字符串（string）的连接。它的运算符是句点（.），使用方法如下：

```
$s1 = 'Hello, ';
$s2 = 'world!';
$greeting = $s1 . $s2;
```

连接的最后结果是$greeting变量的值为Hello, world!。

由于PHP处理变量的特有方式，因此下面的方法可以达到同样的效果：

```
$greeting = "$s1$s2";
```

这是因为在PHP处理变量时，将放置在双引号中的变量用它们的值进行了替代。但是，推荐使用句点连接字符串的正式方法，并且这种方法使用得更加广泛（这将更明显地看出代码中发生了什么）。

另外一种能够起到连接作用的方法是使用连接赋值运算符：

```
$greeting = 'Hello, ';
$greeting .= 'world!';
```

第二行大致的意思是“为$greeting赋予它当前的值，并且同world!连接”。最后的结果是$greeting的值为Hello, world!。

posting.html脚本向handle_post.php页面发送了几个字符串变量。在这些变量中，逻辑上应该将姓和名连接起来。推荐使用像在表单中那样提供用户姓名分开填写的方式，这种方式应用得更加普遍。另外，这种方式更加有利于将两个部分看作一个人名。在编写PHP脚本的时候请注意对此加以考虑。

⇛ 使用连接

(1) 在文本编辑器或者IDE中新建一个文档，命名为handle_post.php（参看脚本5-2）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Forum Posting</title>
</head>
<body>
```

脚本5-2 该PHP脚本展示了连接——最常见的字符串变量操作，可以将它看作是字符串的加法

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Forum Posting</title>
```

```
7   </head>
8   <body>
9   <?php // 脚本5-2 - handle_post.php
10  /* 脚本从posting.html接收5个值:
11  first_name, last_name, email, posting, submit */
12
13  // 可以在此进行错误处理。
14
15  // 从$_POST数组获取数值:
16  $first_name = $_POST['first_name'];
17  $last_name = $_POST['last_name'];
18  $posting = $_POST['posting'];
19
20  // 创建姓名变量:
21  $name = $first_name . ' ' . $last_name;
22
23  // 打印一条信息:
24  print "<div>Thank you, $name, for your posting:
25  <p>$posting</p></div>";
26
27  ?>
28  </body>
29  </html>
```

(2) 创建初始PHP标签，如果有必要，开启错误管理：

```
<?php // 脚本5-2 - handle_post.php
```

如果没有启用display_errors，或者error_reporting的级别设置错误，请参见第3章，在这里添加适当的代码以改变这些设置。

(3) 将表单中的数据赋值给局部变量：

```
$first_name = $_POST['first_name'];
$last_name = $_POST['last_name'];
$posting = $_POST['posting'];
```

表单使用POST方式进行数据传输，因此所有的表单数据在$_POST中都将可用。

这个示例没有包含Email相关的代码，这是因为用不着它，但是也可以复制这些代码来引用这个值。

(4) 使用连接创建一个新的$name变量：

```
$name = $first_name . ' ' . $last_name;
```

在这里，连接操作将两个变量和一个空格合并在一起，从而创建了一个名为$name的新变量。假设输入Elliott和Smith作为两个人名输入框的值，那么$name的值将是Elliott Smith。

(5) 向用户输出消息：

```
print "<div>Thank you, $name, for your posting:
<p>$posting</p></div>";
```

这条消息向用户报告了在表单中输入的内容。

(6) 关闭PHP代码段并完成HTML页面：

```
?>
</body>
</html>
```

(7) 将脚本保存为`handle_post.php`，并放置在同`posting.html`相同的目录下（在启用了PHP的服务器上），并且在Web浏览器上测试表单和脚本（参见图5-2和图5-3）。

Please complete this form to submit your posting:

First Name: Jeremy

Last Name: Messersmith

Email Address: jm@example.org

Posting: This is my posting. It could be more original.

Send My Posting

图5-2 使用中的HTML表单

Thank you, Jeremy Messersmith, for your posting:

This is my posting. It could be more original.

图5-3 PHP页面运行结果

请注意，必须通过URL来加载表单（http://*something*），这样当提交表单时，用来处理的PHP脚本也会通过URL来运行。

✓提示

- 如果在表单中使用不同类型的引号并且看到打印结果中有多余的斜线，请参看第3章框注“Magic Quotes”中对于这种问题的解释和处理方法。
- 请注意，理解PHP中单引号和双引号的不同是非常重要的。在单引号中的字符将被按照字面处理，而在双引号中的字符将会被解释（interpreted）（例如，一个变量的名称将会被它的值所代替）。请复习第3章中的相关内容。
- 可以按照需要将许多字符串连接起来。甚至还可以将数字连接成为字符串：

```
$new_string = $s1 . $s2 . $number;
```

  这是因为PHP属于弱类型（weakly typed）编程语言，这就意味着它的变量并不会被锁定为一种特定的格式。在这里，`$number`变量将会被转换成为字符串并且附加在`$new_string`变量的值中。
- 连接有很多种使用方式，甚至当为函数提供参数的时候。例如下面这个不常见但却实用的示例：

```
$text = nl2br($heading . $body);
```

  `nl2br()`函数在第1章简要介绍过，5.3节将会详细介绍这个函数。

5.3　处理换行符

字符串中的换行符通常给PHP开发新手带来一些问题。用户可以在textarea表单元素中用敲击Return或者Enter的方式输入多行文本。每次敲击Return或者Enter的结果在字符串中都相当于产生一个换行符。这些换行符在textarea中会起作用，但是在PHP页面呈现中将不会产生任何效果（参见图5-4和图5-5）。

Please complete this form to submit your posting:

First Name: Rocky

Last Name: Votolato

Email Address: rv@example.edu

Posting:
Here's one line.
Here's another line.
Here's a third line.

Send My Posting

图5-4　表单数据中文本区里的换行符

Thank you, Rocky Votolato, for your posting:

Here's one line. Here's another line. Here's a third line.

图5-5　换行符没有被Web浏览器呈现出来

为了能够在Web页面呈现时与换行符有等同的效果，可以使用break标签：
。幸运的是，PHP中的nl2br()函数能够将换行符自动转换成为break标签：

```
$var = nl2br($var);
```

我们将在handle_post.php中应用这个函数，以便用户提交的内容能够保持其原有格式。

⇛ 转换换行符

(1) 如果handle_post.php（参看脚本5-2）不在开启状态的话，在文本编辑器上打开它。

(2) 当为$posting变量赋值时应用nl2br()函数（参看脚本5-3）：

```
$posting = nl2br($_POST['posting']);
```

脚本5-3　通过使用nl2br()函数，Web浏览器正确显示了发布文本区中的换行符

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Forum Posting</title>
7   </head>
8   <body>
9   <?php // 脚本5-3 - handle_post.php #2
10  /* 脚本从posting.html接收5个值：
11  first_name, last_name, email, posting, submit */
```

```
12
13 // 可以在此进行错误处理。
14
15 // 从$_POST数组获取数值:
16 $first_name = $_POST['first_name'];
17 $last_name = $_POST['last_name'];
18 $posting = nl2br($_POST['posting']);
19
20 // 创建姓名变量:
21 $name = $first_name . ' ' . $last_name;
22
23 // 打印一条信息:
24 print "<div>Thank you, $name, for your posting:
25 <p>$posting</p></div>";
26
27 ?>
28 </body>
29 </html>
```


现在`$posting`将被赋值为`$_POST ['posting']`，其中所有的换行符都被转换成为HTML的break标签。

(3) 保存文件，放置在同`posting.html`相同的目录（在启用了PHP的服务器上）中，并且在Web浏览器中进行测试（参见图5-6）。

```
Thank you, Rocky Votolato, for your posting:

Here's one line.

Here's another line.

Here's a third line.
```

图5-6　同图5-4中相同的提交数据现在被正确的显示出来

✓提示

- ❑ 还能够通过在双引号之间放置换行符（`\n`）的方式向字符串中插入换行符。
- ❑ 其他的HTML标签（如<p>标签）同样能够影响Web呈现页面的布局。可以使用一个替换函数将换行符（或其他字符）转换为<p>标签，但是这样做的代码比直接调用`nl2br()`要复杂得多。
- ❑ 字符串中的换行符发送到浏览器后会产生效果，但效果只会出现在HTML的源代码中（参见图5-7）。

```
<div>Thank you, Rocky Votolato, for your posting:
<p>Here's one line.

Here's another line.

Here's a third line.</p></div></body>
</html>
```

图5-7　图5-5页面的HTML源代码显示了Web浏览器中包含换行符的效果（即，在HTML源代码中加入空白）

5.4　HTML 和 PHP

正如本书在此之前多次提及的那样，PHP是一项被频繁用来向Web浏览器发送数据的服务器

端技术。这些数据可以是纯文本、HTML代码或者两者都有的形式。

在本章的主要示例里，数据将从一个HTML表单中输入，然后使用PHP在Web浏览器上打印出来。一个潜在的问题是，用户可以在表单中输入HTML字符，这将会对页面的格式产生影响（参见图5-8和图5-9），或许还会导致更糟的结果，从而引发安全方面的问题。

图5-8　如果用户在发布时输入HTML代码

图5-9　当再次打印时它被Web浏览器呈现出来

可以使用一些PHP函数来处理PHP字符串变量中的HTML标签。

- ❑ `htmlspecialchars()`将特定的HTML标签转换为实体版本。
- ❑ `htmlentities()`将所有的HTML标签转换为实体版本。
- ❑ `strip_tags()`移除所有的HTML和PHP标签。

前两个函数将HTML标签（如`<span>`）转换为实体版本，如`<span>`。这个实体版本在输出时出现但是不被呈现。如果希望显示代码而不发布，可以使用它们其中的任何一个来做到。第3个函数`strip_tags()`用来完全移除所有的HTML和PHP标签。

出于两个原因这里应当查看在用户提供的数据中是否有特殊的标签。首先，正如已经提到的，用户提交的HTML可能会对页面的呈现产生影响（参见图5-10）（例如，表格错位、扭曲CSS样式，或者只是添加不应该出现的格式）。第二个原因更加重要。由于JavaScript是被放置在HTML标签中的，因此怀有恶意的用户会提交JavaScript，当它们重新显示在页面上时，这些脚本会被执行。这就是*跨站点脚本*（cross-site scripting，XSS）攻击的实现方式。

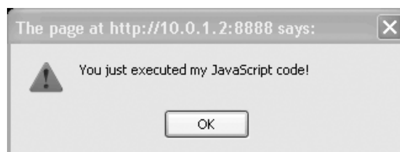


图5-10　在Web浏览器中显示用户提交的HTML会产生很恶劣的后果，如执行JavaScript脚本

为了了解这些函数的影响，下面在对handle_post.php的重写过程中将逐个使用它们并展现各自的结果。

⇛ 处理HTML和PHP

(1) 如果handle_post.php不是开启状态的话，请在文本编辑器或IDE中将它打开（参看脚本5-3）。

(2) 在print行之前添加下面的代码行（参看脚本5-4）：

```
$html_post = htmlentities($_POST ['posting']);
$strip_post = strip_tags($_POST  ['posting']);
```

脚本5-4 这个版本的PHP脚本用两种不同的方式处理HTML标签

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Forum Posting</title>
7   </head>
8   <body>
9   <?php // 脚本5-4 - handle_post.php #3
10  /* 脚本从posting.html接收5个值：
11  first_name, last_name, email, posting, submit */
12
13  // 可以在此进行错误处理。
14
15  // 从$_POST数组获取数值：
16  $first_name = $_POST['first_name'];
17  $last_name = $_POST['last_name'];
18  $posting = nl2br($_POST['posting']);
19
20  // 创建姓名变量：
21  $name = $first_name . ' ' . $last_name;
22
23  // 处理变量中的HTML标签：
24  $html_post = htmlentities($_POST ['posting']);
25  $strip_post = strip_tags($_POST ['posting']);
26
27  // 打印一条信息：
28  print "<div>Thank you, $name, for your posting:
29  <p>Original: $posting</p>
30  <p>Entity: $html_post</p>
31  <p>Stripped: $strip_post</p></div>";
32
33  ?>
34  </body>
35  </html>
```

为了搞清楚这两个函数之间的区别，将它们都应用于posting，并在过程中创建两个新的变量。本书在这里使用$_POST['posting']而不是$posting，是因为$posting已经反映了

nl2br()函数的应用，这意味着有可能引用用户没有显式输入的break标签。

(3) 将print语句按照下面的方法改写：

```
print "<div>Thank you, $name, for your posting:
<p>Original: $posting</p>
<p>Entity: $html_post</p>
<p>Stripped: $strip_post</p></div>";
```

为了突出结果的不同，打印出3个不同的posting版本。第一个是与输入时相同的原始posting，第二个是对posting应用了htmlentities()函数后的版本。它将显示HTML标签而不呈现它们的作用。最后一种是使用了strip_tags()的版本，结果中没有任何HTML（或PHP）标签。

(4) 保存文件，将其放在同posting.html相同的目录下（在启用了PHP的服务器上），并且在Web浏览器上再次进行测试（参见图5-11和图5-12）。

Please complete this form to submit your posting:

First Name: Laura

Last Name: Burhenn

Email Address: lb@example.com

Posting: I don't understand why it says something.

Send My Posting

图5-11 作为posting的一部分所输入的HTML字符将被PHP处理

Thank you, Laura Burhenn, for your posting:

Original: I don't understand why it says *something*.

Entity: I don't understand why it says something.

Stripped: I don't understand why it says something.

图5-12 PHP结果页显示了没有被修改过的原始的posting，以及应用了htmlentities()和strip_tags()后的效果

如果查看结果PHP页面的HTML源代码（参见图5-13），还将看到应用这些函数后的效果。

```
<p>Original: I don't understand why it says <em>something</em>.</p>
<p>Entity: I don't understand why it says &lt;em&gt;something&lt;/em&gt;.</p>
<p>Stripped: I don't understand why it says something.</p></div></body>
```

图5-13 图为图5-10中所示内容的HTML源代码

✓提示

- ❑ 出于安全的考虑，用htmlentities()、htmlspecialchars()或者strip_tags()来处理需要在Web浏览器上打印的用户提供的任何数据，通常情况下这是很好的做法。因为本书的目的是最小化可能带来的复杂度，因此这里没有这样做。
- ❑ html_entity_decode()函数同htmlentities()函数正好相反，它将HTML实体转换为相应的HTML代码。
- ❑ 另外一个向Web浏览器输出字符串的函数是wordwrap()。这个函数按照指定长度折行处

理字符串。

- 为了将换行符转换为break标签并且同时移除所有HTML或者PHP标签，可以在strip_tags()函数后使用nl2br()：

```
$posting = nl2br(strip_tags($_POST['posting']));
```

  在这行代码中，strip_tags()函数将被首先调用，并且它的结果会被发送给nl2br()函数。

5.5　字符串的编码和解码

第3章的3.6节中展示了如何使用GET方法将数据附加在URL中向页面传送。在示例中，没有使用真正的表单，而是将数据直接加在URL上，通过URL发送到接收的脚本。准确地说，这种方式只能传送一个单词，且不能带有空格和标点。但是如果希望传送多个词的变量值或特殊字符时怎么办呢？

urlencode()函数可以通过URL安全地将任意值传送到PHP脚本。顾名思义，这个函数接受一个字符串，并对之*编码*（encode）（改变它的格式），以便它完全适合作为URL的一部分传输。这个函数用加号（+）替换掉空格，并且将特殊字符（如省略号）转换为较少出现问题的形式。可以编写以下代码来使用这个函数：

```
$string = urlencode($string);
```

为了展示urlencode()函数的应用，我们重写handle_post.php页，添加一个链接，用来向第3个页面传送用户的姓名和Email地址。

⇛ 使用urlencode()

(1) 如果handle_post.php不在开启状态的话，请在文本编辑器或者IDE中打开它（参看脚本5-4）。

(2) 删除在之前那些步骤中添加的htmlentities()和strip_tags()代码行（参看脚本5-5）。

脚本5-5　该脚本在将两个变量添加进链接之前先对它们进行编码。通过使用这种方法，变量被成功地发送给其他页面

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Forum Posting</title>
7   </head>
8   <body>
9   <?php // 脚本5-5 - handle_post.php #4
10  /* 脚本从posting.html接收5个值:
11  first_name, last_name, email, posting, submit */
```

```
12
13  // 可以在此进行错误处理。
14
15  // 从$_POST数组获取数值:
16  $first_name = $_POST['first_name'];
17  $last_name = $_POST['last_name'];
18  $posting = nl2br($_POST['posting']);
19
20  // 创建姓名变量:
21  $name = $first_name . ' ' . $last_name;
22
23  // 打印一条信息:
24  print "<div>Thank you, $name, for your posting:
25  <p>$posting</p></div>";
26
27  // 创建连接另一页面的链接:
28  $name = urlencode($name);
29  $email = urlencode($_POST['email']);
30  print "<p>Click <a href=\"thanks.php?
    name=$name&email=$email \">here</a> to continue.</p>";
31
32  ?>
33  </body>
34  </html>
```

(3) 恢复到调用print代码的较老版本。

```
print "<div>Thank you, $name, for your posting:
<p>$posting</p></div>";
```

(4) 在print语句后，添加下面的代码：

```
$name = urlencode($name);
$email = urlencode($_POST['email']);
```

该脚本将会向第2个页面传送这两个变量。因此，这两个变量必须经过编码。

因为之前还没有引用或者使用过$email变量，第二行代码不仅从$_POST数组中检索email的值，而且还在这一步中对它进行了编码。这同下面两行分开的代码执行效果是一样的：

```
$email = $_POST['email'];
$email = urlencode($email);
```

(5) 添加另外一个创建链接的print语句：

```
print "<p>Click <a href=\"thanks.php ?name=$name&email=$email\">
→here </a> to continue.</p>";
```

print语句的核心意图是在这个Web页面中创建一个HTML链接，其源代码将和下面的代码类似：

```
<a href="thanks.php?name=Larry+ Ullman&email=larry%40example.com">here</a>
```

为了实现这个目标，需要硬编码大部分HTML，并且包含适当的变量名称。由于HTML代码需要将链接的URL放置在双引号中，并且print语句已经使用了双引号，因此必须对它们进行转义以便将它们打印出来。

(6) 保存文件，将其放置在启用了PHP的服务器的适当目录下，并且在Web浏览器上测试（参见图5-14和图5-15）。

Please complete this form to submit your posting:

First Name: Christopher

Last Name: O'Reilly

Email Address: chris.oreilly@example.com

Posting: Nothing like a piano cover of Radiohead or Elliott Smith!

Send My Posting

图5-14　表单的另一次应用

Thank you, Christopher O'Reilly, for your posting:

Nothing like a piano cover of Radiohead or Elliott Smith!

Click here to continue.

图5-15　现在处理脚本显示了到另外一个页面的链接

也请注意，点击这个链接将导致一个服务器错误，因为还没有编写thanks.php脚本。

(7) 查看处理页面的HTML源代码以查看在HTML代码中的结果链接（参见图5-16）。

```
<p>Click <a href="thanks.php?name=Christopher+O%27Reilly&email=chris.oreilly%40example.com">here</a>
```

图15-16　图5-15的源代码，显示动态生成的链接

✓提示

- ❑ 从表单直接发送的值，在被发送之前会自动进行URL−编码，接收脚本收到后再对其进行解码。因此，你只需要使用urlencode()函数手动编码数据即可（如本节例子所示）。
- ❑ urldecode()函数同urlencode()正好相反，它接受一个编码过的URL并且将之转换回标准形式。它的使用并不会那么频繁，这是因为PHP能够自动将它获取到的大部分值进行解码。
- ❑ 因为可以对函数使用连接，所以新的print语句可以按照下面的方法进行编写：

```
print 'Click <a href="thanks.php? name=' . $name .
→'&email=' . $email . '">here</a> to continue.';
```

  这种方法有两个额外的好处：首先它使用单引号标识语句的开始和结束，这就意味着不用做双引号的转义操作；其次，使用的变量更加显而易见，它们不会淹没在一大堆其他的代码中。
- ❑ 不用为了在URL中使用数值的PHP值而对其进行编码，因为它们并不包含会出现问题的字符。尽管如此，对它们进行编码也不会造成什么影响。
- ❑ 在本章的末尾，我会提示你创建thanks.php，这是个感谢页面，会显示用户的名字和Email地址。

Thank you, Christopher O'Reilly.
We will contact you at chris.oreilly@example.com.

图5-17　这是第3个页面，会在本章结尾时创建，该页面打印从URL中接收到的信息

字符串的加密和解密

为了保护数据，开发人员常常对数据进行*加密*（encrypt），改变它们的形态，将之转换为几乎不可能看明白的形式。密码就是通常希望进行加密的一个例子。取决于所期望建立的安全级别，用户名、Email地址和电话号码也都可以加密。

可以使用`crypt()`函数来对数据进行加密，但是请注意没有解密选项可用（它是一种*单向*加密方式）。因此，可以用它来对密码进行加密并且保存，但是无法确定解码值。在Web应用程序中使用这个函数，可以在注册的时候对用户密码进行加密，然后当用户登录时，他们输入的密码同样会被加密，并且会对密码的两个加密版本进行对比。`crypt()`的语法如下：

```
$data = crypt($data);
```

第二种加密函数是`mcrypt_encrypt()`，可以使用名为`mcrypt_decrypt()`的函数对它进行解密。麻烦的是，为了能够使用这两种函数，Mcrypt扩展必须同PHP模块一同安装。它的用法和语法自然更加复杂（在我编著的*PHP 5 Advanced: Visual QuickPro Guide*[Peachpit Press, 2007]一书中有相关内容的讨论）。

如果数据被存储在数据库中，也可以使用数据库应用程序（例如MySQL、PostgreSQL、Oracle或者SQL Server）中的内置函数来执行加密和解密。大多数技术都提供单向或者双向的加密工具，但还取决于你所使用的技术。

5.6　查找子字符串

PHP有一些函数可以用来拆解、搜索以及比较字符串。尽管这些函数通常都要在一定条件下使用，我们将在第6章中讨论这些条件，但是这些函数是非常重要的，本书在这里一定要对它们进行介绍，第6章会更加正式地运用它们。

在本章的较前部分已经学习到如何连接字符串。除了用短小的字符串片段组成更大的字符串外，还可以从字符串中提取出某一部分。使用方法提取部分字符串的技巧是，必须对字符串本身有所了解，以便能够有效地执行这样的操作。

`strtok()`函数使用一个预定的分隔符作为标记（例如，逗号或者空格），从大字符串中创建子字符串。例如，如果用户在一个字段中输入他们的全名（假定他们的姓和名使用空格进行分隔），可以通过这样的代码确定他们的名：

```
$first = strtok($_POST['name'], ' ');
```

这行代码告诉PHP从`$_POST['name']`中获取直到遇到空格之前的所有内容。

如果你得到的用户以“Surname，First”这种格式输入的全名，你需要通过下面的代码找到他的姓：

```
$last = strtok($_POST['name'], ', ');
```

第二种提取部分字符串的方法是引用字符串中字符的*索引位置*（indexed position）。字符串的*索引*（index）是一个字符的位置值，从字符串开头数起。但是，如同大多数编程语言一样，PHP所有的索引都是以0开始的。例如，字符串Larry的索引中，L在位置0上，接下来的a的索引为1，r为2，第二个r为3，y为4。虽然字符串Larry的长度为5个字符，但是它的索引是从0到4的（也就是索引通常是从0到字符串长度减1）。

了解这些后，可以根据子字符串中字符所在的索引位置，利用`substr()`函数来创建子字符串，例如：

```
$sub = substr($string, 0, 10);
```

第一个参数是需要提取子字符串的主字符串。第二个参数指明子字符串从哪里开始提取，也就是它在主字符串中的索引位置（0的意思是希望从第一个字符开始）。第三个参数的意思是从起始点开始，需要为子字符串提取的字符数量（10）。如果主字符串不够10个字符的长度，子字符串将以主字符串的结尾作为结束。这个参数是可选的，如果省略，子字符串也将以主字符串的结尾作为结束。

还可以使用负数来从字符串末尾处开始计数：

```
$string = 'ardvark';
$sub = substr($string, -3, 3); // ark
```

第二行的意思是，返回的结果是从字符串的末尾处倒数第3个字符开始的3个字符。就本例来说，省略第3个参数也可以得到相同的结果：

```
$sub = substr($string, -3); // ark
```

可以使用`strlen()`来了解字符串中字符的数量：

```
print strlen('Hello, world!'); // 13
```

该函数在计算数量的时候将包括空格和标点符号。而`str_word_count()`函数是用来获取字符串中*单词*数量的。它同`substr()`一起，将用在handle_post.php脚本的下一个版本中。

字符串比较

为了比较两个字符串，通常可以使用等号运算符，这将在第6章中进行介绍。另外，还可以使用这些函数：

- `strcmp()`将用返回一个整数的方式返回两个字符串的比较结果；
- `strnatcmp()`类似，但是从语言学上更加精确。

还有不区分大小写的比较函数：`strcasecmp()`和`strnatcasecmp()`。

查看一个字符串是否包含另外一个字符串（难度等同于在干草堆里找一根针），可以使用下面的函数：

- `strstr()`返回字符串中从被找寻的字符串第一次出现的位置开始直到字符串结束的所有字符；

- strpos()返回所查找字符串在主字符串中第一次出现的位置。

这两个函数都有各自不区分大小写的对应函数：stristr()和stripos()。每个函数通常都用来在一个条件语句中判断是否找到子字符串。

⇒ 创建子字符串

(1) 如果handle_post.php是未开启状态，在文本编辑器或IDE中打开它（参看脚本5-5）。

(2) 在print语句之前，添加下面的代码（参看脚本5-6）：

```
$words = str_word_count($posting);
```

脚本5-6　该版本的handle_post.php返回发布内容中的单词数量，并且限制显示的内容仅为开头的50个字符

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Forum Posting</title>
7   </head>
8   <body>
9   <?php // 脚本5-6 - handle_post.php #5
10  /* 脚本从posting.html接收5个值:
11  first_name, last_name, email, posting, submit */
12
13  // 可以在此进行错误处理。
14
15  // 从$_POST数组获取数值。
16  $first_name = $_POST['first_name'];
17  $last_name = $_POST['last_name'];
18  $posting = nl2br($_POST['posting']);
19
20  // 创建姓名变量:
21  $name = $first_name . ' ' . $last_name;
22
23  // 计算单词数量:
24  $words = str_word_count($posting);
25
26  // 将发送的内容截断一部分:
27  $posting = substr($posting, 0, 50);
28
29  // 打印一条信息:
30  print "<div>Thank you, $name, for your posting:
31  <p>$posting...</p>
32  <p>($words words)</p></div>";
33
34  ?>
35  </body>
36  </html>
```

在这个版本的脚本中，希望对用户发布的内容做两件事情。一个是显示内容包含的单词数量。这个信息在这里收集，然后赋值给$words变量。

(3) 在下一行代码中（同样也在print语句之前）添加如下代码：

```
$posting = substr($posting, 0, 50);
```

这里希望脚本做的第二件事是限制所显示的内容为开头的50个字符。可以这样应用这个示例，例如，如果一个页面显示发布内容的开始部分，然后提供给用户通往完整内容的链接。为了执行显示限制，这里调用substr()函数。

(4) 依照下面的代码更新print语句：

```
print "<div>Thank you, $name, for your posting:
<p>$posting...</p>
<p>($words words)</p></div>";
```

这里有两处变化。首先在posting后添加了省略号，这表明这只是整个posting内容的一部分。然后在另外一段中打印单词的数量。

(5) 删除urlencode()和相应的print两行代码。

这里指的是在之前的脚本中添加的链接thanks.php脚本的代码。

(6) 保存文件，将其放在启用了PHP的服务器上适当的目录中，并且再次在Web浏览器中测试（参见图5-18和图5-19）。

图5-18　发布的内容多于50个字符

图5-19　显示的内容将被剪短，单词的数量也会被显示出来

✓提示

- 如果想检验一个字符串是否匹配某些特定的格式（如是否是有效的Email地址），需要使用正则表达式。正则表达式（regular expression）是一种高级概念，可以用它来定义模式并且检验某个值是否符合这种模式。参见PHP手册或我编写的《PHP 6与MySQL 5基础教程》，了解更多信息。

5.7　替换局部字符串

我们不仅需要在一个字符串中查找子字符串（如本章之前部分所讨论的那样），还可能需要用新的值替换子字符串（replace substring）。可以使用str_ireplace()函数来实现：

```
$string = str_ireplace($needle, $replacement, $haystack);
```

每次$needle在$haystack出现时，这个函数都将之替换为$replacement。例如：

```
$me = 'Larry E. Ullman';
$me = str_ireplace('E.', 'Edward', $me);
```

现在变量$me的值变为Larry Edward Ullman。

该函数执行的搜索不区分大小写。为了增加限制性，可以使用str_replace()进行区分大小写的搜索。在下一个示例中，将使用str_ireplace()来在提交的文本中除去“脏话”。

这里介绍与字符串有关的最后一个函数trim()。这个函数用来移除在字符串首尾处的所有空白：空格、换行符和制表符。在一个字符串变量中有多余的空格是非常常见的，这或许是由于用户输入信息时的不小心，或者由于不正规的HTML代码造成。出于简洁、数据完整性和Web设计的考虑，需要在使用这些字符串之前删除空白。向Web浏览器发送多余的空白会使页面看起来古怪，如果是发送给数据库或者Cookie，终将在日后造成令人遗憾的结果（例如，如果密码中含有多余的空白，那么当输入不带有这些空白的密码时将不会被正确匹配）。

调整字符串的大小写

少数PHP的函数能够用来变换字符串字母的大小写。

- ucfirst()将字符串的第一个字母变成大写。
- ucwords()将字符串中每个单词的第一个字母变成大写。
- strtoupper()将整个字符串都变为大写。
- strtolower()将整个字符串都变为小写。

请注意，因为全球的人名各式各样，对于PHP（或者任何编程语言）来说的确没有完美的方法来对人名进行自动格式化。事实上，我对于改变用户提供数据的大小写状态非常犹豫，除非有真正好的理由这样做。

trim()函数自动从字符串的开头和结尾处（而不是中间）去除所有的空白。trim()的使用格式如下：

```
$string = ' extra space before and after text ';
$string = trim($string);
// $string 现在等于 'extra space before and after text'.
```

⇒ 使用str_ireplace()

(1) 如果handle_post.php是未开启状态，在文本编辑器或IDE中打开它（参看脚本5-6）。

(2) 对表单数据应用trim()函数（参看脚本5-7）：

```
$first_name = trim($_POST['first_name']);
$last_name = trim($_POST['last_name']);
$posting = trim($_POST['posting']);
```

脚本5-7 这是对脚本进行处理的最后一个版本，应用了trim()函数，并且用一些X来代替脏话

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Forum Posting</title>
7   </head>
8   <body>
9   <?php // 脚本5-7 - handle_post.php #6
10  /* 脚本从posting.html接收5个值:
11  first_name, last_name, email, posting, submit */
12
13  // 可以在此进行错误处理。
14
15  // 从$_POST数组获取数值。
16  // 使用trim()去掉多余的空格:
17  $first_name = trim($_POST['first_name']);
18  $last_name = trim($_POST['last_name']);
19  $posting = trim($_POST['posting']);
20
21  // 创建姓名变量:
22  $name = $first_name . ' ' . $last_name;
23
24  // 计算单词数量:
25  $words = str_word_count($posting);
26
27  // 去掉脏话:
28  $posting = str_ireplace('badword', 'XXXXX', $posting);
29
30  // 打印一条信息:
31  print "<div>Thank you, $name, for your posting:
32  <p>$posting</p>
33  <p>($words words)</p></div>";
34
35  ?>
36  </body>
37  </html>
```

只有当收到的数据的首尾包含多余的空白，trim()才被应用。

(3) 删除使用substr()的代码段。

```
$posting = substr($posting, 0, 50);
```

这里希望在这个示例中看到完整的发布内容，因此删除了对substr()的调用。

(4) 在print语句之前，添加下面的代码：

```
$posting = str_ireplace('badword', 'XXXXX', $posting);
```

这个特殊的示例找出了在发布内容中的脏话并将它们去除。这里的代码用`badword`来代替实际中的脏话（当然可以使用任何你想使用的词语）。

如果希望替换更多的脏话，可以使用下面的多行代码来实现：

```
$posting = str_ireplace('badword1', 'XXXXX', $posting);
$posting = str_ireplace('badword2', 'XXXXX', $posting);
$posting = str_ireplace('badword3',  'XXXXX', $posting);
```

(5) 更新`print`语句以便它不再使用省略号：

```
print "<div>Thank you, $name, for your posting:
<p>$posting</p>
<p>($words words)</p></div>";
```

(6) 保存文件，将其放在启用了PHP的服务器的适当目录下，并在Web浏览器上再次进行测试（参见图5-20和图5-21）。

图5-20　如果用户输入了不希望他们使用的词

图5-21　那么可以让PHP来替换它们

✓提示

- ❑ `str_ireplace()`函数将捕捉到内容中的脏话。例如，如果输入I feel like using badwords，结果将会是I feel like using XXXXXs。
- ❑ `str_ireplace()`函数还能够接受一个子字符串的数组、一个替换字符串的数组，甚至一个主字符串的数组。因为你也许此时并不知道数组是什么，所以在这里将不对这个技术进行解释。
- ❑ 如果需要去除一个字符串开头或结尾的空白，而不是两者都需要去除，PHP将`trim()`函数分为两个更加特殊的函数：`rtrim()`移除字符串变量结尾处的空白，而`ltrim()`则处理字符串开头的空白。两者一起用的效果等同于`trim()`。

```
$string = rtrim($string);
$string = ltrim($string);
```

5.8　回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

5.8.1　回顾

- 如何创建一个字符串？
- 单引号与双引号在使用上有什么不同？
- 什么是连接操作符？什么是连接赋值操作符？
- 将包括换行符的字符串输出到浏览器会有什么影响？如何将换行符转换成br标签。
- 如果表单元素的值会在浏览器中显示，而在该表单元素中输入的是HTML可能导致什么问题？此时有什么办法处理提交的表单数据？
- 哪个函数可以使数据安全地包含在URL中传递？
- 如何去除字符串中的问题字符？如果不去除问题字符会出现什么问题？
- 字符串中字符索引的起始值是什么？
- `trim()`函数的功能是什么？

5.8.2　实践

- 在PHP手册中查找一个本章介绍的新函数，利用页面中的链接查看其他字符串相关函数。
- 在PHP手册中查找`substr()`函数，并查看该函数的其他示例，加深理解`substr()`的使用方法。
- 编写脚本5-5中要用到的`thanks.php`脚本。如果需要帮助，可以参考第3章中的`hello.php`脚本（参见脚本3-7）。
- 重写`handle_post.php`（参见脚本5-7）最终版中的`print`语句，使用单引号和连接代替双引号。
- 创建一个新的HTML表单，接收字符串值。接下来，创建一个PHP脚本，接收表单中的数据，处理任意HTML或PHP代码，以某些方式操作数据，并打印出结果。

第 6 章

控制结构

6

本章内容

- ❑ 创建HTML表单
- ❑ if条件语句
- ❑ 验证函数
- ❑ 使用else
- ❑ 更多运算符
- ❑ 使用elseif
- ❑ switch条件语句
- ❑ for循环
- ❑ 回顾和实践

控制结构是指条件和循环，它是编程语言的主要组成部分。PHP包含两种常见的条件控制语句：if和switch，本章将介绍如何使用它们。你可以使用条件控制语句来测试某个变量或表达式是否满足某一特定的条件，并基于测试的结果执行相应的操作。这些功能能够用来构建更具动态性的Web网站。

讨论if条件的讨论将引入最后的两类运算符：比较运算符和逻辑运算符（在之前的章节中已经介绍了算数运算符和赋值运算符）。通常这些运算符还将连同布尔概念中的真/假概念一起运用。

最后，本章将开始在编程中使用循环，它可以以一定的迭代次数重复同一操作。循环能够节省很多编程的时间，在第7章中，还将看到它同数组联合使用获得更为丰富的功能。

6.1 创建 HTML 表单

如同在之前的章节中那样，本章的示例将基于一个向PHP页面发送数据的HTML表单。在这个例子中，表单是一个简单的注册页面，它需要以下信息（参见图6-1）：

- ❑ Email地址；
- ❑ 密码；
- ❑ 密码确认；

- 生日（用于验证年龄）；
- 喜欢的颜色（用于定制外观）；
- 网站条款协议（常见需求）。

Please complete this form to register:

Email Address:

Password:

Confirm Password:

Year You Were Born: YYYY

Favorite Color: Pick One

I agree to the terms (whatever they may be).

Register

图6-1　本章中使用的HTML表单

接下来的步骤将在涉及PHP代码之前先处理表单。

⇛ 创建HTML表单

(1) 在文本编辑器或者IDE中新建一个HTML文档，命名为register.html（参看脚本6-1）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Registration Form</title>
</head>
<body>
<!-- 脚本6-1 - register.html -->
<div><p>Please complete this form to register:</p>
```

脚本6-1　这个虚构的注册表单是本章的示例基础

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Registration Form</title>
7   </head>
8   <body>
9   <!-- 脚本6-1 - register.html -->
10  <div><p>Please complete this form to register:</p>
11
12  <form action="handle_reg.php" method="post">
13
14    <p>Email Address: <input type="text" name="email" size="30" /></p>
```

```
15
16   <p>Password: <input type="password" name="password" size="20" /></p>
17
18   <p>Confirm Password: <input type="password" name="confirm" size="20" /></p>
19
20   <p>Year You Were Born: <input type="text" name="year"
     value="YYYY" size="4" /></p>
21
22   <p>Favorite Color:
23   <select name="color">
24   <option value="">Pick One</option>
25   <option value="red">Red</option>
26   <option value="yellow">Yellow</option>
27   <option value="green">Green</option>
28   <option value="blue">Blue</option>
29   </select></p>
30
31   <p><input type="checkbox" name="terms" value="Yes" /> I agree
     to the terms (whatever they may be).</p>
32
33   <input type="submit" name="submit" value="Register" />
34
35 </form>
36
37 </div>
38 </body>
39 </html>
```

(2) 创建表单的起始标签：

```
<form action="handle_reg.php" method="post">
```

如同之前的许多示例那样，该页选用POST方法。用来处理的脚本是handle_reg.php，它被action属性标识。

(3) 为Email和密码创建输入框：

```
<p>Email Address: <input type="text" name="email" size="30" /></p>
<p>Password: <input type="password" name="password" size="20" /></p>
<p>Confirm Password: <input type="password" name="confirm" size="20" /></p>
```

这些代码行的意义不言而喻。每行都使用HTML的<p></p>标签换行来增加Web浏览器中的间距。请注意，创建了两个密码输入框——第二个输入框用来确认第一个框中的输入内容。输入框的类型为密码，就不会暴露用户输入的内容（参见图6-2），因此它是一种良好的保密方式，让用户再次输入（以便用户对输入的密码进行再次确认）。

(4) 为生日创建输入框：

```
<p>Year You Were Born: <input type="text" name="year"
→value="YYYY" size="4" /></p>
```

这里不使用下拉菜单来显示50年或者100年的选项，而是让用户直接将他们的出生年份填写

在文本框中。可以在文本框中预先设置value属性来指明该文本框所需的正确年份格式（参见图6-1）。

(5) 创建下拉菜单，用户可以在下拉菜单中选择喜欢的颜色：

```
<p>Favorite Color:
<select name="color">
<option value="">Pick One</option>
<option value="red">Red</option>
<option value="yellow">Yellow</option>
<option value="green">Green</option>
<option value="blue">Blue</option>
</select></p>
```

事实上，添加这个下拉框的目的在于，它可以在本章后面的一些特殊示例中使用，但是在这里它可以用来自定义用户登录后站点的外观。你可以按照自己的意愿添加尽可能多的颜色。

(6) 创建网站条款协议之前的同意复选框：

```
<p><input type="checkbox"name="terms" value="Yes" />
I agree to the terms (whatever they may be).</p>
```

许多网站一般使用复选框让用户接受一些条款或许可协议。这个表单没有连接到许可条款页面的链接，不过这无关紧要，没有人会去看那些条款（不管怎么说，这个表单只是一个示例）。总之，这里是要说明PHP脚本是如何处理复选框的。

(7) 添加一个提交按钮并且关闭表单：

```
  <input type="submit" name="submit" value="Register" />
</form>
```

(8) 完成HTML页面：

```
</div>
</body>
</html>
```

(9) 将文件保存为register.html，放在启用了PHP的服务器上适当的目录下，并且在Web浏览器中加载该页面。

✓提示

- 注册页面经常需要用户对他们的密码进行确认，并且有时候还需要确认他们的用户名和Email地址（无论在登录时需要什么信息）。
- 大多数注册页面使用昵称或者Email地址作为用户名，如果使用Email地址作为用户名，这将让用户能够更好地记住他们的注册信息（一个用户可能只有有限几个Email地址，但是对于不同的网站可能有好多不同的用户名）。此外，Email地址本质上就是唯一的，而用户名不是唯一的。

6.2　if 条件语句

编程基本的条件语句是标准的if（也曾被称作为if-then条件语句，现在then被默认而不提出）。这种条件语句的语法非常简单：

```
if (condition) {
        statement(s);
}
```

该条件必须同圆括号一同使用，并且语句部分要被放置在花括号中，这些是需要执行的命令（例如，打印字符串或者将两数相加）。每句单独的语句（或者命令）都必须拥有自己的分号以示本行的结束，但是对于和一个条件相关联的语句数量并没有限制。

开发人员通常将这些语句从if起始的代码行开始进行缩进处理，用来指出它们是条件的结果，但是这种格式并不是语法上要求的。也有人这样使用语法：

```
if (condition)
{
        statement(s);
}
```

如何安排花括号是个人喜好问题——这在网上有一些小争议。选择一种你喜欢的风格，并坚持使用就可以了。

在每行语句后没有使用分号，忘记一个打开或者关闭的圆括号或者花括号，或者在花括号后使用了分号都将导致错误发生。因此在代码中使用条件语句时请留意语法是否正确。

PHP使用布尔值（TRUE和FALSE）来检测是否执行该语句。如果条件为TRUE，则执行语句；如果是FALSE，则不执行（参见图6-2）。

通过本章的学习（本章的大部分），将开发出一个PHP脚本，用来完全验证register.html表单数据。在开始阶段，该脚本的第一个版本将只创建这个验证过程的基本框架，定义和使用值为布尔类型的变量，它将跟踪验证过程的成败。

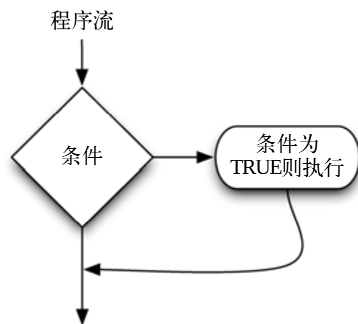


图6-2　IF条件语句控制脚本程序流的方式

⇒ 创建一个if条件语句

(1) 在文本编辑器或者IDE中新建一个文档，命名为handle_reg.php（参看脚本6-2）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Registration</title>
```

```
</head>
<body>
<h1>Registration Results</h1>
```

脚本6-2 将构建一个PHP脚本的框架，用来验证表单数据

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Registration</title>
7   </head>
8   <body>
9   <h1>Registration Results</h2>
10  <?php // 脚本6-2 - handle_reg.php
11  /* 脚本从register.html接收8个值：
12  email, password, confirm, year, terms, color, submit */
13
14  // 可以在此进行错误处理。
15
16  // 创建标记变量，标记表单状态：
17  $okay = TRUE;
18
19  // 如果没有错误，打印一条正确消息：
20  if ($okay) {
21    print '<p>You have been successfully registered (but not really).</p>';
22  }
23  ?>
24  </body>
25  </html>
```

(2) 开始PHP脚本部分，如果需要添加错误管理：

```
<?php // 脚本6-2 - handle_reg.php
```

如果没有开启display_errors，或者如果error_reporting被设置了错误的级别，请参见第3章，在这里添加适当的代码以改变这些设置。

(3) 创建一个标记变量：

```
$okay = TRUE;
```

验证表单数据，需要使用一个标记变量，用它表示表单是否已经填写完毕。这里称之为“标记”变量是因为它会存储一个简单的值，用来指明事物的状态。例如，yes表示表单填写完毕；no表示表单没有填写完毕。

这个变量被初始化为布尔值中的TRUE，意思是假定表单已正确填写完毕。布尔类型不区分大小写，因此在这里可以书写为True或者true。

(4) 如果一切就绪，那么打印一条消息：

```
if ($okay) {
  print '<p>You have been successfully registered (but not really).</p>';
}
```

通过本章的学习，验证例程将被添加进这段脚本中，用来检验提交的表单数据。如果有任何数据在通过例程检验时失败，那么`$okay`将被设定为FALSE。在这种情况下，该条件语句的判断结果也会是FALSE，因此消息不会被打印出来。但是，如果数据通过了每个例程的检验，那么`$okay`将为TRUE，在这种情况下消息就会被打印出来。

(5) 完成PHP代码片段和HTML页面：

```
?>
</body>
</html>
```

(6) 将文件保存为`handle_reg.php`，放在启用了PHP的服务器适当的目录下（同`register.html`在同一目录下），并且在Web浏览器中对它们进行测试（参见图6-3和图6-4）。

图6-3　填写HTML表单（不必填写完整）

图6-4　结果如此

当然，事实上该脚本将总是打印成功的消息，因为没有什么让`$okay`的值为FALSE。甚至可以直接运行脚本并且查看到相同的结果。

✓提示

- 如果条件的语句段部分只有一行，从技术上来说可以不使用花括号。这种情况下，可以将条件写成以下两种格式：

  ```
  if (condition) statement;
  ```

  或

  ```
  if (condition)
      statement;
  ```

  可以以这种格式正确运行。但是，本书建议最好一直使用多行加花括号的格式（如同在语法介绍中示例的那样）来增加一致性并减少错误。

6.3　验证函数

PHP有许多常用的函数来验证表单数据。本章的示例将用到其中3个重要的函数。

第一个是empty()函数，它用来检验是否一个给定的变量拥有除0和空字符串之外的其他值。当变量没有值时（或者值为0或者一个空字符串），它将返回TRUE，反之则返回FALSE：

```
$var1 = 0;
$var2 = 'something';
$var3 = ' '; // 一个空字符串
empty($var); // TRUE，没有已定义的值
empty($var1); // TRUE，空值
empty($var2); // TRUE，非空值
empty($var3); // TRUE，空值
```

这个函数用于确认表单中的文本框是否被填写。例如，如果有一个叫做Email的文本框，并且用户在提交表单前并没有在其中做任何输入，那么$_POST['email']变量存在，但是它的值是空值。

下一个是isset()函数，它同empty()几乎相反，虽然区别非常小。当变量拥有值（包括0、FALSE或者空字符串）时，isset()函数返回TRUE，反之则返回FALSE。例如：

```
$var1 = 0;
$var2 = 'something';
$var3 = ' '; // 一个空字符串
isset($var); // FALSE，没有已定义的值
isset($var1); // TRUE
isset($var2); // TRUE
isset($var3); // TRUE
```

isset()函数通常用来验证无文本的表单元素，如多选钮、单选钮和选择菜单。

最后是is_numeric()函数，它在提交的变量是一个有效的数字类型的值时返回TRUE，反之返回FALSE。正数、小数，甚至字符串（如果它们是有效的数字）都可以通过is_numeric()的验证：

```
$var1 = 2309;
$var2 = '80.23';
$var3 = 'Bears';
is_numeric($var1); // TRUE
is_numeric($var2); // TRUE
is_numeric($var3); // FALSE
```

让我们开始在PHP脚本中应用这些函数，以便验证一些实际数据。

⇒ 验证表单数据

(1) 如果handle_reg.php不在开启状态，在文本编辑器或者IDE中打开它（参看脚本6-2）。

(2) 在文档的head部分，定义一个CSS类（参看脚本6-3）：

```
<style type="text/css" media="screen">
  .error { color: red; }
</style>
```

脚本6-3　通过使用if条件语句和empty()函数，这个PHP脚本检验了是否提供了Email地址和密码的值

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Registration</title>
7     <style type="text/css" media="screen">
8       .error { color: red; }
9     </style>
10  </head>
11  <body>
12  <h1>Registration Results</h1>
13  <?php // 脚本6-3 - handle_reg.php #2
14  /* 脚本从register.html接收7个值：
15  email, password, confirm, year, terms, color, submit */
16
17  // 可以在此进行错误处理。
18
19  // 创建标记变量，标记表单状态：
20  $okay = TRUE;
21
22  // 验证Email地址：
23  if (empty($_POST['email'])) {
24    print '<p class="error">Please enter your email address.</p>';
25    $okay = FALSE;
26  }
27
28  // 验证密码：
29  if (empty($_POST['password'])) {
30    print '<p class="error">Please enter your password.</p>';
31    $okay = FALSE;
32  }
33
34  // 如果没有错误，打印一条正确消息：
35  if ($okay) {
36    print '<p>You have been successfully registered (but not really).</p>';
37  }
38  ?>
39  </body>
40  </html>
```

这个CSS类将被用来格式化打印出的注册错误信息。

(3) 验证Email地址：

```
if (empty($_POST['email'])) {
  print '<p class="error">Please enter your email address.</p>';
  $okay = FALSE;
}
```

这个if条件语句使用代码empty($_POST['email'])作为它的条件。如果这个变量是空

的，意味着它没有值、有一个为0的值，或者值为空字符串，那么条件语句的结果是TRUE。在这种情况下，将执行print语句，并且$okay变量将被赋值为FALSE（标志着不是所有的输入信息都正确）。

如果变量不为空，那么条件语句的结果就为FALSE，将永远不会调用print函数，并且$okay将保持它原有的值。

(4) 对密码做同样的验证：

```
if (empty($_POST['password'])) {
  print '<p class="error">Please enter your password.</p>';
  $okay = FALSE;
}
```

这是对Email验证的重复，只有变量名与print语句进行了相应的更改。其余的表单输入也将被及时验证。

所有打印出来的错误信息都被放置在HTML中class值为error的<p>标签里。这样做是为了应用CSS的格式（也就是说错误信息将用红色的字体打印，而不是书上显示的颜色（黑白））。

(5) 将文件保存为handle_reg.php，并放置在与register.html相同的目录下（在启用了PHP的服务器上），并且在Web浏览器上测试表单和脚本（参见图6-5和图6-6）。

图6-5 如果遗漏了表单输入中的Email地址或者密码

图6-6 将看到这样的信息

(6) 再次以不同完成程度提交表单以测试更多的结果。

如果提供了Email地址和密码的值，结果将同图6-4非常相似，这是因为$okay变量的值仍然为TRUE。

✓提示

- 当在条件语句中使用函数时，就像这里使用的empty()那样，非常容易因为遗忘关闭圆括号而看到错误信息。请在编写任何控制结构时对所使用的语法特别注意。
- isset()函数的一个用法是用来避免引用不存在的变量。如果PHP被设置为显示通知（参见第3章），那么诸如使用未被声明的变量$var将会导致错误出现。可以通过编写下面的代码避免这样的错误发生：

```
if (isset ($var)) {
```

```
  // 可以对$var做任何想做的事情。
}
```

- 即便是向PHP脚本发送的所有表单数据都使用字符串的方式，is_numeric()函数仍然能够用于表单中的数据，这是因为它处理只包含数字的字符串。
- isset()函数能够接受任意数量的变量作为参数：

```
if (isset($var1, $var2)) {
  print 'Both variables exist.';
}
```

如果所有被命名的变量都被设置，该函数将返回TURE，如果有任何变量未被设置，函数将返回FALSE。

6.4　使用 else

在if条件语句之后要介绍的下一个逻辑形式是if-else条件语句。它可以用来构建这样的条件：当条件满足时执行一组语句，而条件不满足时执行另外的语句：

```
if (condition) {
        statement(s);
} else {
        other_statement(s);
}
```

使用这个结构需要牢记的要点是，除非完全满足条件，否则将执行else语句。换句话说，else后的语句指明的是默认行为，因此在if条件后的语句才是规则的例外（参见图6-7）。

让我们重写handle_reg.php页，使用if-else条件语句把对生日数据的验证并入其中。在这个过程中，将创建一个新的变量来表示生日。

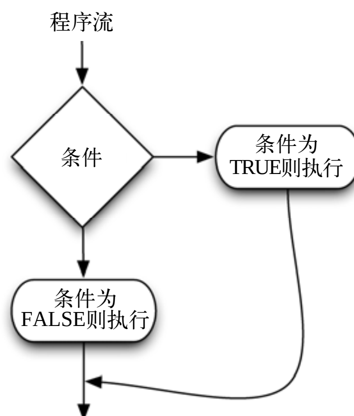


图6-7　IF-ELSE条件语句控制脚本中程序流的方式

⇒ 使用else的步骤

(1) 如果handle_reg.php不在开启状态的话，在文本编辑器或IDE中打开它（参看脚本6-3）。

(2) 在$okay条件之前密码验证之后创建一个新的条件语句（参看脚本6-4）：

```
if (is_numeric($_POST['year'])) {
```

脚本6-4　添加if-else条件语句后，可以对生日的日期进行验证并且在这个过程中创建一个新的变量

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html;
      charset=utf-8"/>
6     <title>Registration</title>
7     <style type="text/css" media="screen">
8       .error { color: red; }
9     </style>
10  </head>
11  <body>
12  <h1>Registration Results</h1>
13  <?php // 脚本6-4 - handle_reg.php #3
14  /* 脚本从register.html接收7个值：
15  email, password, confirm, year, terms, color, submit */
16
17  // 可以在此进行错误处理。
18
19  // 创建标记变量，标记表单状态：
20  $okay = TRUE;
21
22  // 验证Email地址：
23  if (empty($_POST['email'])) {
24    print '<p class="error">Please enter your email address.</p>';
25    $okay = FALSE;
26  }
27
28  // 验证密码：
29  if (empty($_POST['password'])) {
30    print '<p class="error">Please enter your password.</p>';
31    $okay = FALSE;
32  }
33
34  // 验证出生年：
35  if (is_numeric($_POST['year'])) {
36    $age = 2011 - $_POST['year']; // 计算年龄。
37  } else {
38    print '<p class="error">Please enter the year you were born as
      four digits.</p>';
39    $okay = FALSE;
40  }
41
42  // 如果没有错误，打印一条正确消息：
43  if ($okay) {
44    print '<p>You have been successfully registered (but not really).</p>';
45    print "<p>You will turn $age this year.</p>";
46  }
47  ?>
48  </body>
49  </html>
```

由于year变量的值应当是一个数字，因此可以使用is_numeric()替代empty()函数来验证它的值。这只是表单元素验证的第一步，后面的脚本会进一步验证表单元素。

(3) 创建一个新的变量：

```
$age = 2011 - $_POST['year'];
```

如果$_POST['year']变量含有一个数值（这就意味着条件为TRUE），那么$age变量将被赋值为当前年份减去用户提交的年份。由于还没有介绍日期函数，这里在等式中硬编码当前年份。

(4) 加入一个else子句：

```
} else {
  print '<p class="error">Please enter
  →the year you were born as four digits.</p>';
  $okay = FALSE;
}
```

如果year没有含有数值，那么将会打印错误信息，并且$okay变量的值将被设置为FALSE（任何验证过程失败都将是这样的情况）。

(5) 在最终的print语句后，$okay的条件语句内，打印$age的值：

```
print "<p>You will turn $age this year.</p>";
```

如果$okay变量的值仍为TRUE，那么说明提交的数据已经通过所有的验证。这意味着已经计算过用户的年龄了（表明用户今年有多大岁数），并将结果打印出来。

(6) 将脚本保存在与register.html相同的目录中（在启用了PHP的服务器上），并在Web浏览器上再次测试（参见图6-8、图6-9和图6-10）。

Please complete this form to register:
Email Address: me@example.com
Password: ••••••••
Confirm Password: ••••••••
Year You Were Born: YYYY
Favorite Color: Green
I agree to the terms (whatever they may be).
Register

图6-8　再次对表单进行测试，忽略一些年份信息

Registration Results
Please enter the year you were born as four digits.

图6-9　如果年份没有含有数值，将打印这样的错误消息

Registration Results
You have been successfully registered (but not really).
You will turn 57 this year.

图6-10　如果用户正确提交了出生年份（数值），脚本会计算出用户的年龄并打印出来(假设用户同时提交了Email地址和密码)

✓提示

- 另外一个好用的验证函数是checkdate()，可以用来确认某个日期是否存在（或者在过去是否存在）。可以这样进行使用：

```
if (checkdate($month, $day, $year)) {...
```

6.5　更多运算符

前面的章节已经对PHP中大部分的运算符以及使用它们的变量的类型做了讨论。这些运算符包括对数字的算术运算符：加（+）减（-）乘（*）除（/），值增加和减少1的快捷方式：自增（++）和自减（--），以及赋值运算符（=），用来为变量设置值，不考虑类型。我们已经学习了连接符（.），它用来将一个字符串附加给另外的字符串。

这些运算符都用来对变量的值进行处理，但是它们在条件语句中却用得比较少。现在将开始

探索比较运算符和逻辑运算符，它们也很重要。表6-1列举了PHP中更加全面的运算符种类。

表6-1　PHP中的运算符

运　算　符	用　　法	类　　型
+	加	算术
-	减	算术
*	乘	算术
/	除	算术
%	模（除法的余数）	算术
++	增量	算术
--	减量	算术
=	将值赋给变量	赋值
==	相等	比较
!=	不相等	比较
<	小于	比较
>	大于	比较
<=	小于或等于	比较
>=	大于或等于	比较
!	取反	逻辑
AND	与	逻辑
&&	与	逻辑
OR	或	逻辑
||	或	逻辑
XOR	或非	逻辑
.	连接	字符串

6.5.1　比较运算符

当赋值运算符（等号）第一次在第2章中介绍时，我们已经认识到它的含义同之前按照常规理解的含义有所不同。代码行：

```
$var = 5;
```

并不代表$var等于5，而是说它被赋值为5。这是非常重要的差别。

当编写条件语句时，将经常希望了解某个变量是否等于某个特定的值（同用户名或者密码匹配，等等），这就不能只使用等号（这是因为它是用来赋值而不是表示同某值相等）。出于这种目的，需要使用相等运算符（==），它用两个等号来表示相等关系：

```
$var = 5;
if ($var == 5) { ...
```

这两行代码一同使用将首先把$var的值设置为5，然后当$var的值等于5时条件的结果为

TRUE。这个示例再次解释了PHP中等号用法的重要不同，以及必须将赋值运算符和比较运算符仔细区分开的原因。

下一个是比较运算符：不等于，它由一个惊叹号和一个等号组成（!=）。此外，剩下的比较运算符和它们在数学中使用的是相同的：小于（<）、大于（>）、小于等于（<=）和大于等于（>=）。

作为对比较运算符的示范，将对用户的生日年份是否在2011年之前进行确认，并且还要确认密码是否同原始密码匹配。

⇒ 使用比较运算符

(1) 如果handle_reg.php不在开启状态的话，在文本编辑器或者IDE中打开它（参看脚本6-4）。

(2) 在密码验证之后，检查两个密码是否匹配（参看脚本6-5）：

```
if ($_POST['password'] != $_POST['confirm']) {
  print '<p class="error">Your confirmed password does not
  →match the original password. </p>';
  $okay = FALSE;
}
```

脚本6-5　页面处理脚本的这个版本使用比较运算符以进行对密码和year值的验证

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Registration</title>
7     <style type="text/css" media="screen">
8       .error { color: red; }
9     </style>
10  </head>
11  <body>
12  <h1>Registration Results</h1>
13  <?php // 脚本6-5 - handle_reg.php #4
14  /* 脚本从register.html接收7个值:
15  email, password, confirm, year, terms, color, submit */
16
17  // 可以在此进行错误处理。
18
19  // 创建标记变量，标记表单状态:
20  $okay = TRUE;
21
22  // 验证Email地址:
23  if (empty($_POST['email'])) {
24    print '<p class="error">Please enter your email address.</p>';
25    $okay = FALSE;
26  }
27
28  // 验证密码:
29  if (empty($_POST['password'])) {
```

```
30    print '<p class="error">Please enter your password.</p>';
31    $okay = FALSE;
32  }
33
34  // 检查两个密码是否相等:
35  if ($_POST['password'] != $_POST['confirm']) {
36    print '<p class="error">Your confirmed password does not match
      the original password.</p>';
37    $okay = FALSE;
38  }
39
40  // 验证出生年:
41  if (is_numeric($_POST['year'])) {
42    $age = 2011 - $_POST['year']; // Calculate age this year.
43  } else {
44    print '<p class="error">Please enter the year you were born as four
      digits.</p>';
45    $okay = FALSE;
46  }
47
48  // 检查出生年是否在当前年之前:
49  if ($_POST['year'] >= 2011) {
50    print '<p class="error">Either you entered your birth year wrong or
      you come from the future!</p>';
51    $okay = FALSE;
52  }
53
54  // 如果没有错误，打印一条正确消息:
55  if ($okay) {
56    print '<p>You have been successfully registered (but not really).</p>';
57    print "<p>You will turn $age this year.</p>";
58  }
59
60  ?>
61  </body>
62  </html>
```

为了比较这两个字符串的值，可以使用不等于运算符。你也可以使用一个字符串比较函数（参见第5章），不过这已经足够了。

(3) 在year验证之后，检查year的值是否在2011年之前：

```
if ($_POST['year'] >= 2011) {
  print '<p class="error">Either you entered your birth year
  →wrong or you come from the future!</p>';
  $okay = FALSE;
}
```

如果用户键入的出生年份为2011或者在其之后，将会有错误显示（可以根据实际时间修改年份）。

(4) 将脚本保存在与register.html相同的目录中（在启用了PHP的服务器上），并且在Web浏览器中再次测试（参见图6-11和图6-12）。

6

Please complete this form to register:

Email Address: me@example.com

Password: ························

Confirm Password: ····

Year You Were Born: 2023

Favorite Color: Green

I agree to the terms (whatever they may be).

Register

图6-11　再次运行表单

Registration Results

Your confirmed password does not match the original password.

Either you entered your birth year wrong or you come from the future!

图6-12　两个新的验证检查

✓提示

- 在比较两个来自于表单的字符串值(如密码和密码确认)之前,建议对它们两个使用trim()函数，以去除多余的空格。本书在这里没有这样做的原因是避免将问题处理得过于复杂，但是推荐保持这个良好的习惯。
- 另外一个检查文本输入框已经被填充的方法如下所示(而不是使用empty()函数),例如:

```
if (strlen ($var) > 0 ) {
  // $var正确。
}
```

- 在一个if条件语句中，如果在编写$var == 5时错误地写为$var = 5，将看到相应的条件语句总是会执行。这是因为尽管条件$var == 5也许是或者也许不是TRUE，但条件$var = 5的结果都将是TRUE。
- 一些程序员提倡反向编写条件语句，如：

```
if (5 == $var) {
```

虽然这样看起来很怪，但是当你不经意间编写出如5 = $var这样的代码，就会出现一个错误（更容易捕获这个错误），因为数字5不能被赋值给其他的值。

6.5.2　逻辑运算符

使用PHP编写的条件能够对TRUE或者FALSE状态进行识别。正如已经看到的那样，这可以使用函数和比较运算符很好地实现。逻辑运算符是我们在本章中要讨论的最后一种运算符类型，它可以用来创建更多的复杂或者显而易见的结构。

举一个在PHP中关于TRUE条件的简单例子，用来确认一个变量的值不为零或者空字符串或者FALSE，代码如下：

```
$var = 5;
if ($var) { ...
```

我们已经看到在进行处理的PHP脚本中使用了$okay变量。

下面使用逻辑运算符表达的条件结果也是TRUE：

```
if (5 >= 3) { ...
```

当引用没有值的变量（或者值为0或一个空字符串）或创建的结构不合逻辑时，条件结果将为FALSE。下面的条件结果将是FALSE：

```
if (5 <= 3) { ...
```

在PHP中，感叹号（!）是NOT运算符。可以用来对语句的TRUE/FALSE状态进行转化。例如：

```
$var = 'value';
if ($var) {... // TRUE
if (!$var) {... // FALSE
if (isset($var)){... // TRUE
if (!isset($var)){... // FALSE
if (!empty($var)){... // TRUE
```

除了这些简单的部分条件，PHP还提供了5种逻辑运算符：两个版本的与（AND和&&）、两个版本的或（OR和||，或使用两个称为管道的字符）、以及异或（XOR）。当一个运算符有两种选择时（例如，使用与和或时），它们仅仅只有优先级上的区别。在几乎所有的情况中，都可以交替地使用与或者或的任何一种版本。

使用圆括号和逻辑运算符可以创建更加复杂的if条件语句。对于一个AND条件语句来说，每一个组成部分都必须是TRUE，这样才能保证整个条件的结果为TRUE。对于OR来说，至少有一个子句是TRUE才能保证整个条件的结果是TRUE。以下的条件均为TRUE：

```
if ( (5 <= 3) OR (5 >= 3) ) { ...
if ( (5 > 3) AND (5 < 10) ) { ...
```

以下的条件均为FALSE：

```
if ( (5 != 5) AND (5 > 3) ) { ...
if ( (5 != 5) OR (5 < 3) ) { ...
```

在构造自己的条件语句时，请记住两件重要的事情：首先，为了让符合条件结果的语句能够执行，条件的值必须为TRUE；第二，使用圆括号时，可以忽略优先级的规则，并强制运算符按照所制定的顺序进行处理。

为了举例说明逻辑运算符，我们向handle_reg.php页面添加了两个条件语句，还可以将year条件语句嵌套进其他的条件语句中（参看框注“嵌套条件”以了解更多内容）。

嵌套条件

除了使用逻辑运算符来创建更加复杂的条件语句外，还可以使用嵌套来达到相同的目的（将一个控制结构放置在另外一个控制结构中的过程）。这样做的关键点是将内部条件放置在外部条件的代码段语句部分。例如：

```
if (condition1) {
        if (condition2) {
                statement(s)2;
```

```
        } else { // 条件2
else

other_statement(s)2;
       } // 条件2结束
} else { // 不满足条件1，则：
       other_statement(s)1;
} // 条件1结束
```

正如在本示例中看到的那样，大量使用缩进和注释可以降低结构的复杂度。PHP对嵌套的层数没有规定，只要每个条件的语法正确即可，不管是使用else子句，或者甚至子条件语句是主条件语句的if或者else块中的一部分。

⇒ 使用逻辑运算符

(1) 如果handle_reg.php不在开启状态的话，在文本编辑器或者IDE中打开它（参看脚本6-5）。

(2) 删除已经存在的year验证部分（参看脚本6-6）。

脚本6-6　这里处理的PHP脚本已经被修改为对于year验证的同时使用了多个条件语句和嵌套条件语句的方式。同时还会验证是否勾选注册协议的复选框

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Registration</title>
7     <style type="text/css" media="screen">
8       .error { color: red; }
9     </style>
10  </head>
11  <body>
12  <h1>Registration Results</h1>
13  <?php // 脚本6-6 - handle_reg.php #5
14  /* 脚本从register.html接收7个值:
15  email, password, confirm, year, terms, color, submit */
16
17  // 可以在此进行错误处理。
18
19  // 创建标记变量，标记表单状态:
20  $okay = TRUE;
21
22  // 验证Email地址:
23  if (empty($_POST['email'])) {
24    print '<p class="error">Please enter your email address.</p>';
25    $okay = FALSE;
```

```
26 }
27
28 // 验证密码:
29 if (empty($_POST['password'])) {
30   print '<p class="error">Please enter your password.</p>';
31   $okay = FALSE;
32 }
33
34 // 检查两个密码是否相等:
35 if ($_POST['password'] != $_POST['confirm']) {
36   print '<p class="error">Your confirmed
     password does not match the original
     password.</p>';
37   $okay = FALSE;
38 }
39
40 // 验证年:
41 if ( is_numeric($_POST['year']) AND (strlen($_POST['year']) == 4) ) {
42
43   // 检查出生年是否在2011年之前。
44   if ($_POST['year'] < 2011) {
45     $age = 2011 - $_POST['year']; // 计算年龄。
46   } else {
47     print '<p class="error">Either you entered your birth year
       wrong or you come from the future!</p>';
48     $okay = FALSE;
49   } // 条件2结束。
50
51 } else { // 不满足条件1,则:
52
53   print '<p class="error">Please enter the year you were born as
     four digits.</p>';
54   $okay = FALSE;
55
56 } // 条件1结束。
57
58 // 验证用户是否选中条款:
59 if (!isset($_POST['terms'])) {
60   print '<p class="error">You must accept the terms.</p>';
61   $okay = FALSE;
62 }
63
64 // 如果没有错误,打印一条正确消息:
65 if ($okay) {
66   print '<p>You have been successfully registered (but not really).</p>';
67   print "<p>You will turn $age this year.</p>";
68 }
69 ?>
70 </body>
71 </html>
```

6

我们已经完整地将这些条件重写为嵌套条件，因此最好是将这些旧版本全部删除。

(3) 检查year变量是否为一个4位的数字：

```
if ( is_numeric($_POST['year']) AND (strlen($_POST['year']) == 4) ) {
```

这个条件由两部分组成。第一个部分我们已经看到，是用来测试值是否为有效的数字。第二个部分用来获取year变量的长度（使用strlen()函数），并且检查其值是否为4。由于使用的是AND运算，因此条件每个部分的结果都为TRUE时整个条件才能为TRUE。

(4) 创建一个子条件来验证year的值不在2011之后：

```
if ($_POST['year'] < 2011) {$age = 2011 - $_POST['year'];
} else {
  print '<p class="error">Either
  →you entered your birth year
  →wrong or you come from the
  →future!</p>';
  $okay = FALSE;
} // 条件2结束。
```

这个if-else条件句在主条件句中充当语句部分，只有在条件为TRUE时才运行这部分。这个if-else用来检查year变量是否大于等于2011（即，用户的生日必须早于当前年份）。如果条件为TRUE，就说明用户年龄已经计算完成；否则会打印一条错误信息，并且$okay变量将被设置为FALSE（指明有问题出现）。

注意，这个条件语句与前一个相反：验证值小于某个值，而不是大于或等于某个值。

(5) 完成year的主条件句：

```
} else { // 不满足条件1，则：
  print '<p class="error">Please enter the year you were born
  →as four digits.</p>';
  $okay = FALSE;
} // 条件1结束。
```

else子句完成了在第(3)步中开始的条件语句。如果至少有一个条件为FALSE，那么就会打印信息，并且$okay被设置为FALSE。

(6) 确认用户选中条款协议复选框：

```
if (!isset($_POST['terms'])) {
  print '<p class="error">You must accept the terms.</p>';
  $okay = FALSE;
}
```

如果没有设置$_POST['terms']变量，说明用户没有选中复选框，那么就会打印错误消息。条件语句更加准确的写法如下：

```
if ( !isset($_POST['terms']) AND ($_POST['terms'] == 'Yes') ) {
```

(7) 这是对脚本的唯一变更之处，将脚本再次保存并放置在与register.html相同的目录中（在启用了PHP的服务器上），并且在Web浏览器中再次测试（参见图6-13和图6-14）。

(8) 如果需要，可以将year的值改成将来的年份并且再次提交表单（参见图6-15）。

Please complete this form to register:

Email Address: me@example.com

Password: ••••••••

Confirm Password: ••••••••

Year You Were Born: 15

Favorite Color: Green

I agree to the terms (whatever they may be).

Register

图6-13 PHP脚本遇到不为4位数字的年份时会捕获错误，提交本图填写的表单就会出现这种情况

Registration Results

Please enter the year you were born as four digits.

You must accept the terms.

图6-14 如果没有正确填写字段或没有选中协议复选框，将会打印出错误信息

Registration Results

Either you entered your birth year wrong or you come from the future!

图6-15 `year`验证检查日期是否在2011年之前

✓提示

- 在本书中一直保持着另一个常见的编程惯例，即书写术语TRUE和FALSE时使用全大写方式，虽然这不是PHP必需的。例如，下面的条件是TRUE：

  ```
  if (true) {...
  ```

- 在处理长而复杂的条件时非常容易忘记开启或者关闭的圆括号或花括号，这将导致错误信息出现或者意想不到的结果发生。寻找一个有助于让代码变得清晰的方式（例如，分隔条件语句并使用注释）。另一项实用的技术是先创建完整的条件结构，然后再添加细节部分。
- 如果执行`if-else`语句时有问题，将变量的值打印出来将能帮助调试问题。条件不为TRUE时，是因为变量的值不是预计的那样。

6.6 使用 `elseif`

`if-elseif`（或者`if-elseif-else`）条件同`if-else`条件类似。它的作用就像是运行`if`语句，并且能够按照需要进行任意长度的扩展：

```
if (condition1) {
        statement(s);
} elseif (condition2) {
        other_statement(s);
}
```

这是另外一个示例（参见图6-16）：

```
if (condition1) {
        statement(s);
} elseif (condition2) {
```

6

```
        other_statement(s);
} else {
        other_other_statement(s);
}
```

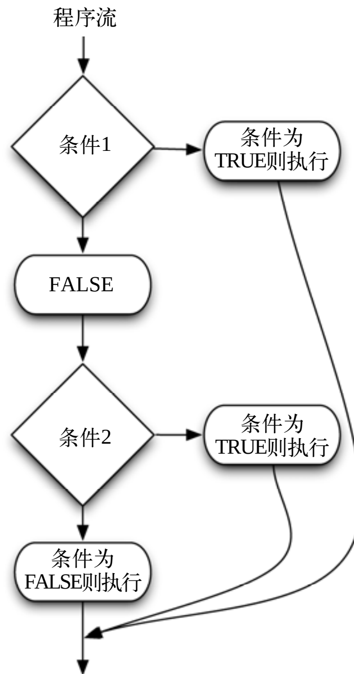


图6-16　IF-ELSEIF-ELSE控制脚本中程序流的方式

必须永远让else位于条件语句的最后一部分，因为只有到这里没有满足任何一个条件才会执行它（或者说，else代表了默认的行为）。但是，也可以将elseif用作if条件中的一部分并且继续多次使用。

作为示例，我们创建一个条件语句，基于所选color值来打印消息。

⇒ 使用elseif的步骤

(1) 如果handle_reg.php不在开启状态的话，在文本编辑器或者IDE中打开它（参看脚本6-6）。

(2) 在$okay条件前，创建一个新的条件语句（参看脚本6-7）：

```
if ($_POST['color'] == 'red') {$color_type = 'primary';
```

脚本6-7　这个多行的if-elseif-else条件打印了关于color的特定信息，并且验证提交的color的值是否为允许的值

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4  <head>
5    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6    <title>Registration</title>
7    <style type="text/css" media="screen">
8      .error { color: red; }
9    </style>
10 </head>
11 <body>
12 <h1>Registration Results</h1>
13 <?php // 脚本6-7 - handle_reg.php #6
14 /* 脚本从register.html接收7个值:
15 email, password, confirm, year, terms, color, submit */
16
17 // 可以在此进行错误处理。
18
19 // 创建标记变量，标记表单状态:
20 $okay = TRUE;
21
22 // 验证Email地址:
23 if (empty($_POST['email'])) {
24   print '<p class="error">Please enter your email address.</p>';
25   $okay = FALSE;
26 }
27
28 // 验证密码:
29 if (empty($_POST['password'])) {
30   print '<p class="error">Please enter your password.</p>';
31   $okay = FALSE;
32 }
33
34 // 检查两个密码是否相等:
35 if ($_POST['password'] != $_POST['confirm']) {
36   print '<p class="error">Your confirmed
     password does not match the original
     password.</p>';
37   $okay = FALSE;
38 }
39
40 // 验证年:
41 if ( is_numeric($_POST['year']) AND (strlen($_POST['year']) == 4) ) {
42
43   // 检查出生年是否在2011年之前。
44   if ($_POST['year'] < 2011) {
45     $age = 2011 - $_POST['year']; // 计算年龄。
46   } else {
47     print '<p class="error">Either you entered your birth year wrong or
       you come from the future!</p>';
48     $okay = FALSE;
49   } // 条件2结束。
50
51 } else { // 不满足条件1，则:
52
53   print '<p class="error">Please enter the year you were born as four
```

6

```
       digits.</p>';
54     $okay = FALSE;
55
56   } // 条件1结束。
57
58   // 验证用户是否选中条款:
59   if ( !isset($_POST['terms']) AND ($_POST['terms'] == 'Yes') ) {
60     print '<p class="error">You must accept the terms.</p>';
61     $okay = FALSE;
62   }
63
64   // 验证颜色:
65   if ($_POST['color'] == 'red') {
66     $color_type = 'primary';
67   } elseif ($_POST['color'] == 'yellow') {
68     $color_type = 'primary';
69   } elseif ($_POST['color'] == 'green') {
70     $color_type = 'secondary';
71   } elseif ($_POST['color'] == 'blue') {
72     $color_type = 'primary';
73   } else { // 有问题!
74     print '<p class="error">Please select your favorite color.</p>';
75     $okay = FALSE;
76   }
77
78   // 如果没有错误，打印一条正确消息:
79   if ($okay) {
80     print '<p>You have been successfully registered (but not really).</p>';
81     print "<p>You will turn $age this year.</p>";
82     print "<p>Your favorite color is a $color_type color.</p>";
83   }
84   ?>
85   </body>
86   </html>
```

color的值来自于一个下拉菜单，它有4个可能的选项：red、yellow、green和blue。这个条件语句将打印出一条信息，重复用户对color的选择，并且对color进行格式化。第一个条件检验是否$_POST['color']的值同字符串red相等。

在条件语句中，一定要使用*相等运算符*（两个等号），而不是*赋值运算符*（一个等号）。

(3) 为第二个color添加一个elseif子句：

```
} elseif ($_POST['color'] == 'yellow') {
  $color_type = 'primary';
```

这个elseif继续完成在第二步中创建的主条件语句。该条件本身是对第二步中条件的重复，只不过使用了一种新的color。

(4) 为另外两种颜色添加elseif子句：

```
} elseif ($_POST['color'] == 'green') {
  $color_type = 'secondary';
} elseif ($_POST['color'] == 'blue') {
  $color_type = 'primary';
```

一旦理解了主要的概念，接下来就只是对每种可能的color值进行重复编写elseif子句的操作了。

(5) 添加一个else子句：

```
} else {
  print '<p class="error">Please select your favorite color.</p>';
  $okay = FALSE;
}
```

如果用户没有选择color，或者处理表单的时候提交了color其他的值（red、yellow、green或blue之外的值），这些条件的结果都将不为TURE，这意味着这个else子句将起作用。它将打印一个错误并且将$okay赋值为FALSE，提示有问题发生。

color的检查顺序无关紧要,只要将else子句放在最后即可。

(6) 在$okay条件语句中，打印用户喜欢的color类型：

```
print "<p>Your favorite color is a $color_type color.</p>";
```

(7) 将脚本保存在与register.html相同的目录中（在启用了PHP的服务器上），然后在Web浏览器上用不同的color选项进行测试（参见图6-17和图6-18）。

Registration Results

You have been successfully registered (but not really).

You will turn 12 this year.

Your favorite color is a secondary color.

图6-17 现在脚本打印出一条信息，告知用户对color的选择

Registration Results

Please select your favorite color.

图6-18 对color选择失败导致出现这条错误信息

✓提示

- ❑ 有个问题很多初学者都不会意识到，对黑客来说，无需使用你准备好的HTML表单就可以将数据提交到PHP脚本，实际上，这相当容易就能做到。因此，验证是否包含所需变量（你所设置的）及其类型和值是非常重要的。
- ❑ PHP也允许将elseif分写为两个单词，如果愿意的话：

```
if (condition1) {
   statement(s);
} else if (condition2) {
   statement(s)2;
}
```

6.7 switch 条件语句

一旦使用了更长的if-elseif-else条件，就会发现使用switch条件能够节省时间，并且

能够让编码更加清晰。switch条件只带有一个可能的条件，通常情况下就只有一个变量：

```
switch ($var) {
        case value1:
                statement(s)1;
                break;
        case value2:
                statement(s)2;
                break;
        default:
                statement(s)3;
                break;
}
```

了解switch条件语句如何运行非常重要，这将决定你是否能够正确地使用它。在switch关键词之后，用一对括号括住一个已定义的变量。PHP会按顺序查看每个case，尽力找到匹配的值。注意，字符串和数值的使用方式遵循PHP规则，数值不需要加引号，字符串则需要用引号括起来。在case value后面是个冒号（不是分号），接下来是相关语句，另起一行缩进。

一旦PHP发现case同条件变量的值相匹配，它将继续执行后续语句。下面是需要重点关注的地方：一旦PHP找到匹配的case，它将继续执行switch语句，直到switch条件语句的末尾（关闭的花括号）或者遇到break语句，直接在此处退出switch结构。因此，使用break语句关闭每个case非常重要，出于一致性考虑，默认的case也要关闭（参见框注“Break、Exit、Die和Continue”中对这些关键字的详细讨论）。

前面的switch条件其实是下面代码的重写：

```
if ($var == value1) {
        statement(s)1;
} elseif ($variable == value2) {
        statement(s)2;
} else {
        statement(s)3;
}
```

因为switch条件使用$var的值作为它的条件，它首先检查$var的值是否等于*value1*，如果是，那么执行*statement(s)1*，如果不是，它将检查是否$var的值等于*value2*，如果是，执行结果为*statement(s)2*。如果这些条件都不满足，switch条件的默认动作将执行*statement(s)3*。

考虑到这一点，让我们用switch重写color条件语句。

⇒ 使用switch条件

(1) 如果handle_reg.php不在开启状态的话，在文本编辑器或IDE中打开它（参看脚本6-7）。

(2) 删除扩展的color条件（参看脚本6-8）。

脚本6-8　switch条件可以简化复杂的if-elseif条件

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Registration</title>
7     <style type="text/css" media="screen">
8       .error { color: red; }
9     </style>
10  </head>
11  <body>
12  <h1>Registration Results</h1>
13  <?php // 脚本6-8 - handle_reg.php #7
14  /* 脚本从register.html接收7个值:
15  email, password, confirm, year, terms, color, submit */
16
17  // 可以在此进行错误处理。
18
19  // 创建标记变量，标记表单状态:
20  $okay = TRUE;
21
22  // 验证Email地址:
23  if (empty($_POST['email'])) {
24    print '<p class="error">Please enter your email address.</p>';
25    $okay = FALSE;
26  }
27
28  // 验证密码:
29  if (empty($_POST['password'])) {
30    print '<p class="error">Please enter your password.</p>';
31    $okay = FALSE;
32  }
33
34  // 检查两个密码是否相等:
35  if ($_POST['password'] != $_POST['confirm']) {
36    print '<p class="error">Your confirmed password does not match the original
      password.</p>';
37    $okay = FALSE;
38  }
39
40  // 验证年:
41  if ( is_numeric($_POST['year']) AND (strlen($_POST['year']) == 4) ) {
42
43    // 检查出生年是否在2011年之前。
44    if ($_POST['year'] < 2011) {
45      $age = 2011 - $_POST['year']; // 计算年龄。
46    } else {
47      print '<p class="error">Either you entered your birth year wrong or
        you come from the future!</p>';
48      $okay = FALSE;
49    } // 条件2结束。
50
51  } else { // 不满足条件1，则:
52
53    print '<p class="error">Please enter the year you were born as four
      digits.</p>';
```

6

```
54    $okay = FALSE;
55
56  } // 条件1结束。
57
58  // 验证用户是否选中条款:
59  if ( !isset($_POST['terms']) AND ($_POST['terms'] == 'Yes') ) {
60    print '<p class="error">You must accept the terms.</p>';
61    $okay = FALSE;
62  }
63
64  // 验证颜色:
65  switch ($_POST['color']) {
66    case 'red':
67      $color_type = 'primary';
68      break;
69    case 'yellow':
70      $color_type = 'primary';
71      break;
72    case 'green':
73      $color_type = 'secondary';
74      break;
75    case 'blue':
76      $color_type = 'primary';
77      break;
78    default:
79      print '<p class="error">Please select your favorite color.</p>';
80      $okay = FALSE;
81      break;
82  } // switch结束。
83
84  // 如果没有错误，打印一条正确消息:
85  if ($okay) {
86    print '<p>You have been successfully registered (but not really).</p>';
87    print "<p>You will turn $age this year.</p>";
88    print "<p>Your favorite color is a $color_type color.</p>";
89  }
90  ?>
91  </body>
92  </html>
```

(3) 开始编写switch：

```
switch ($_POST['color']) {
```

如之前提及的，switch条件语句只包含一个条件：一个变量的名称。在这种情况下，就是$_POST['color']。

(4) 创建第一个case：

```
case 'red':
  $color_type = 'primary';
  break;
```

第一个case检查是否$_POST['color']值为red，如果是，就执行与之前相同的打印语句。然后添加一个break语句以跳出switch语句。

(5) 为第二个color添加一个case：

```
case 'yellow':
  $color_type = 'primary';
  break;
```

(6) 为剩下的color添加case：

```
case 'green':
  $color_type = 'secondary';
  break;
case 'blue':
  $color_type = 'primary';
  break;
```

(7) 添加一个默认的case并且完成switch语句：

```
  default:
    print '<p class="error">Please
    →select your favorite
    →color.</p>';
    $okay = FALSE;
    break;
} // switch结束。
```

这个默认的case等同于用在原始条件中的else子句。

(8) 将脚本在与register.html相同的目录下保存（在启用了PHP的服务器上），并且在Web浏览器中再次测试（参见图6-19和图6-20）。

Registration Results

You have been successfully registered (but not really).

You will turn 47 this year.

Your favorite color is a primary color.

图6-19 当用户选择了一种颜色后，该处理脚本的显示结果

Registration Results

Please select your favorite color.

图6-20 失败后的结果

✓提示

- 在switch条件中默认的case并不是必需的（可以对它进行设置，以便在值不与case中任何一个明确匹配时不做任何响应），但是如果使用它，就要将它作为最后一个case放置。
- 如果在switch条件中使用字符串作为case的值，那么请注意它是区分大小写的，这就意味着在这种情况下，Value同value并不匹配。

Break、Exit、Die和Continue

PHP包含很多语言结构：虽然是一些非函数的工具，但是它们仍然能够在脚本中发挥作用。第一个是break，在讨论switch时已经示范过。break用来退出当前的结构，例如一个switch、一个if-else条件或一个循环。

同break类似的是continue，它终止循环当前的迭代，在循环中剩下的语句将不会被执行，但是循环的条件会被再次检查以确认是否还应当进入该循环。

exit和die是break更加有效的版本（并且它们意义相同）。这两个语言结构终止执行PHP脚本，而不是跳出当前结构。因此，在exit或者die之后所有的PHP代码将永远不会被执行。对于这个问题，在这些结构之后的任何HTML都永远不会被发送给Web浏览器。你将看到die在访问文件和数据库的代码中使用得最为频繁。Exit通常同header()函数联合使用。

6.8　for 循环

循环是本章中讨论的最后一个控制结构。如同之前建议的那样，可以用循环来反复执行一个代码段。这里可能会希望多次打印某个特定的信息，或者希望打印出数组中的每个值。对于以上的这些情况，以及更多可能的情况，可以使用循环来实现（后面的示例将在第7章中诠释）。

PHP提供3种循环：for、while和foreach。While循环同for非常相似，但是在数据库中检索值或者读取文本文件（在边栏中对此进行了介绍）时使用得更加频繁。foreach同使用数组有关，这将在第7章中介绍。

for循环用来以指定的次数反复执行特定的语句（与while不同，while会一直运行直到条件为FALSE，它们相似但是在概念上仍然有显著的区别）。可以在循环中使用临时计数（dummy）变量以达到这个目的：

```
for (initial expression; condition; closing expression) {
     statement(s);
}
```

initial expression被执行了一次：在第一次调用循环的时候；然后condition用来确定是否要执行statement。每当condition为TRUE，并且statement执行完之后，会执行closing expression（参见图6-21）。

这里有一个循环的简单示例，它用来打印从1到10的数字：

```
for ($v = 1; $v <= 10; $v++) {
       print $v;
}
```

练习for循环，我们扩展注册表单，让用户输入完整的出生日期。for循环可以很容易地在HTML表单中创建下拉菜单。

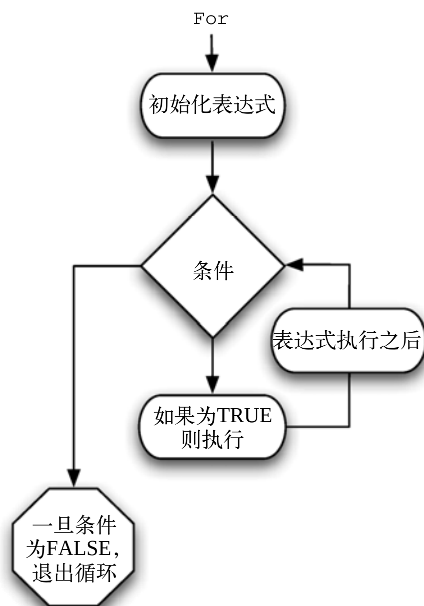


图6-21 这个流程图呈现了在PHP中for循环是如何执行的

6

⇛ 编写一个for循环

(1) 如果register.html不在开启状态的话，在文本编辑器或者IDE中打开它（参见脚本6-1）。

(2) 删除原来的出生年份提示和文本框（参看脚本6-9）。

脚本6-9 这个脚本使用PHP来编写循环以动态地创建month下拉菜单中day的值

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Registration Form</title>
7   </head>
8   <body>
9   <!-- 脚本6-9 - register.php -->
10  <div><p>Please complete this form to register:</p>
11
12  <form action="handle_reg.php" method="post">
13
14    <p>Email Address: <input type="text" name="email" size="30" /></p>
15
16    <p>Password: <input type="password" name="password" size="20" /></p>
17
18    <p>Confirm Password: <input type="password" name="confirm"
```

```
      size="20" /></p>
19
20    <p>Date Of Birth:
21    <select name="month">
22    <option value="">Month</option>
23    <option value="1">January</option>
24    <option value="2">February</option>
25    <option value="3">March</option>
26    <option value="4">April</option>
27    <option value="5">May</option>
28    <option value="6">June</option>
29    <option value="7">July</option>
30    <option value="8">August</option>
31    <option value="9">September</option>
32    <option value="10">October</option>
33    <option value="11">November</option>
34    <option value="12">December</option>
35    </select>
36    <select name="day">
37    <option value="">Day</option>
38    <?php // Print out 31 days:
39    for ($i = 1; $i <= 31; $i++) {
40      print "<option value=\"$i\">$i</option>\n";
41    }
42    ?>
43    </select>
44    <input type="text" name="year" value="YYYY" size="4" /></p>
45
46    <p>Favorite Color:
47    <select name="color">
48    <option value="">Pick One</option>
49    <option value="red">Red</option>
50    <option value="yellow">Yellow</option>
51    <option value="green">Green</option>
52    <option value="blue">Blue</option>
53    </select></p>
54
55    <p><input type="checkbox"
      name="terms" value="Yes" /> I agree to
      the terms (whatever they may be).</p>
56
57    <input type="submit" name="submit" value="Register" />
58
59  </form>
60
61  </div>
62  </body>
63  </html>
```

将原来的一个提示信息变为3个，以表示完整的出生日期：月、日和年。

(3) 将出生日期的提示放在密码确认之后，颜色选择之前，添加文本提示和月份列表：

```
<p>Date Of Birth:
<select name="month">
```

```
<option value="">Month</option>
<option value="1">January</option>
<option value="2">February</option>
<option value="3">March</option>
<option value="4">April</option>
<option value="5">May</option>
<option value="6">June</option>
<option value="7">July</option>
<option value="8">August</option>
<option value="9">September</option>
<option value="10">October</option>
<option value="11">November</option>
<option value="12">December</option>
</select>
```

首先是一个文本提示，告诉用户提供完整的出生日期。接下来是一个选择菜单，用于让用户选择出生月份。选项的值是一个数值，而显示的文本是字符串（每个月的英文单词）。

(4) 创建生日的选择菜单：

```
<select name="day"><option value="">Day</option>
```

代码一开始是select表单元素，用于让用户选择出生日期中的月份。这个列表的值会由PHP生成。

6

(5) 创建一个新的PHP段：

```
<?php
```

因为PHP能够嵌入到HTML中，可以用它填充下拉菜单。 我们从标准的PHP标记开始。

(6) 创建for循环来打印31天作为选择菜单的选项：

```
for ($i = 1; $i <= 31; $i++) {
  print "<option value=\"$i\">$i </option>\n";
}
```

循环从创建一个名为$i的临时计数变量开始。在第一次使用循环时，变量被设置为1。然后，当$i小于等于31时，执行循环体。这些循环体是print行，创建诸如

```
<option value="1">1</option>
```

的代码，后面跟一个换行符（用\n创建）。在循环体执行之后，$i变量的值将会增加1。然后再次检查条件($i<=31)，之后重复该过程。

(7) 关闭PHP代码段：

```
?>
```

(8) 将文件保存为register.php。

现在必须把文件用.php后缀名进行保存，以便PHP代码能够被执行。

(9) 将文件放置在启用了PHP的服务器上适当的目录中，并且在Web浏览器中测试（参见图6-22）。

只要这些脚本与handle_reg.php在同一个目录中，你甚至可以以纯HTML的形式填写并提交表单。

(10) 如果需要看一下生成的PHP选项，可以查看HTML源代码（参见图6-23）。

Please complete this form to register:

Email Address:

Password:

Confirm Password:

Date Of Birth: Month Day YYYY

Favorite Color: Pick One

I agree to the terms (whatever they may be).

Register

图6-22　新版本的HTML表单，有一些自动生成的内容

```
        <option value="11">November</option>
        <option value="12">December</option>
        </select>
        <select name="day">
        <option value="">Day</option>
        <option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
```

图6-23　如果查看表单的HTML源代码，将看到for循环生成的数据

✓提示

- 大家习惯上喜欢使用简单的变量（$i、$j、$k）作为for循环中的计数器。
- 同if条件语句一样，在编写while和for循环时，如果只有一条语句，可以将所有代码写在一行。但是，不推荐这么做。
- 循环能够相互嵌套。可以在循环中放置条件语句，也可以在条件语句中放置循环，等等。
- 请格外关注所编写的循环条件，以便循环能够在某一点上结束。否则，你会创建一个无限循环，脚本将永远地执行下去。

while循环

在PHP中3种循环中的第二种就是while循环，只要是你定义的条件为TRUE，while就会继续执行。同for循环一样，它在每次迭代之前都会对条件进行检查。一旦条件变为FALSE，while循环将会退出：

```
while (condition) {
        statement(s);
}
```

for和while循环的主要区别是，while不包含初始条件，也不执行关闭表达式。

还可以使用do...while循环作为选择，它保证循环体中的语句将至少执行一次（这在while循环中并不是必要的）：

```
do {
        statement(s);
} while (condition);
```

虽然使用两种主要循环结构（while和for）有相当数量的交叠时，在编程时将发现有时一种比另外一种更合理。在从数据库中检索数据时while循环用得更加频繁（参见第12章）。

6.9　回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

6.9.1　回顾

- 为什么在注册过程中让用户再次确认密码非常重要？
- PHP中if条件语句的基本结构是什么？if-else、if-elseif和if-elseif-else条件语句的基本结构是什么？
- empty()和isset()函数的区别是什么？
- 什么是赋值运算符？什么是相等运算符？
- 如果不知道$var变量的任何信息，以下条件语句是TRUE还是FALSE：

```
if ($var = 'donut') {
```

- 以下运算符的含义：
  - &&
  - ||
  - !
- switch条件语句的语法是什么？switch一般用于哪些情况？
- for循环语句的语法是什么？

6.9.2　实践

- 查找PHP手册中各种操作符的用法。
- 重写handle_reg.php脚本，使用变量代替硬编码的值获取当前年份。
- 出于调试目的，在handle_reg.php脚本的起始处加入一些代码，打印接收到的变量的值。提示：有两种方式，一种简单，一种复杂。
- 重写某一版的handle_reg.php脚本，显示用户选择的最喜欢的颜色。提示：可能需要使用CSS和字符串连接。
- 更新handle_reg.php脚本，通过3个表单元素验证用户的生日：month、day和year。创建一个变量，显示用户的生日，格式为：MM/DD/YYYY（需要使用连接）。

第 7 章

使用数组

7

本章内容

- 什么是数组
- 创建数组
- 向数组添加项
- 访问数组元素
- 创建多维数组
- 排序数组
- 字符串和数组之间的转换
- 在表单中创建数组
- 回顾和实践

接下来将要学习数组，也是本书介绍的最后一种变量。数组同数值或者字符串有着显著的不同，并且如果没有掌握和理解它，大多数PHP编程工作都将无法实现。

因为数组独特的特性，和介绍其他变量类型相比，本章将专门放慢速度。从这章开始将全面介绍数组的概念，以及创建和使用数组的基础知识。接下来将介绍多维数组和一些数组相关的函数。本章内容还将包含字符串数组的转化，并解释如何在HTML表单中创建数组。

7.1 什么是数组

数组组成一个复杂但是非常有用的概念。然而数值和字符串都是标量（scalar）变量（这意味着它们只含有一个单独的值），数组是组合到一个重写变量的多个值的集合。一个数组能够由数值和/或字符串（和/或其他数组）组成，因此比起简单的字符串或者数字，数组能够让一个变量以指数级承载更多的信息。例如，如果希望使用字符串来创建一个杂货列表，可能会这样编写代码：

```
$item1 = 'apples';
$item2 = 'bananas';
$item3 = 'oranges';
```

每增加一个项目，就需要创建一个新的字符串。这样做非常麻烦，并且在查询整个列表或者

代码中的任何特定值时会变得非常困难。可以通过将整个列表放置到一个数组（可以命名为$items）中来简化这个问题，该数组将包含所需要的所有项目（参见表7-1）。

表7-1　杂货列表数组

项目编号	项　目
1	apples
2	bananas
3	oranges

使用数组，就可以对列表进行添加、排序、搜索等操作。我们先讲述数组的语法，了解如何操作数组。

数组的语法规则

其他的变量类型——数值和字符串类型——都有一个变量名和相应的值（例如，$first_name可以等于Larry）。数组也有名称，并且使用相同的命名规则，这些规则如下所示：

- 以美元符号开头；
- 紧接开头符号之后为一个字母或者下划线；
- 以字符、数值或者下划线的任意组合结尾。

但是数组的不同之处在于它们拥有多个*元素*（element）（可将表7-1中每行的内容看作为一个元素）。一个元素由一个*索引*（index）或称为*键*（key）（这两个词可以相互替换）及其值组成。在表7-1中，项目编号就是键，而项目为值。

数组的值通过它们的索引被引用。数组能够用数值或者字符串作为它的键（或两者同时），这取决于你如何设置。

一般来说，使用数组看起来同使用任何其他变量一样，除了引用特殊值时需要用方括号（[]）包含它的键。因此，$items代表对整个数组的引用，而$items[1]只是针对数组中特定的元素（在本示例中表示对apple的引用）。

超全局变量

通过这本书的学习，我们已经处理了一些数组：$_SERVER、$_GET和$_POST。这些数组同$_COOKIE、$_SESSION和$_ENV一同都被称为超全局变量。

$_POST数组接受使用POST方法向表单发送的所有数据。它对表单内输入单元的名称进行索引，并且它的值代表了这些表单元素的值。因此，$_POST['name']对使用代码：

```
<input type="text" name="name"/>
```

创建的表单输入单元里键入的值进行引用。

类似地，$_GET引用使用GET方法向表单发送的数据，或者通过URL进行传输的数据。$_COOKIE引用存储在cookie中保存的数据，而$_SESSION引用保存在session中的数据（本书将在第9章中涉及它们）。$_ENV同$_SERVER相似，包含的值与运行PHP的计算机有关。

7

7.2　创建数组

创建数组正式的方法是使用array()函数。它的语法如下所示：

```
$list = array ('apples', 'bananas', 'oranges');
```

除非特别指出，否则数组的索引自动从0开始。本示例并未为元素指定特定的索引，因此apple作为第一项自动索引为0，而第二项索引为1，第三项索引为2。

可以在使用array()时为索引赋值：

```
$list = array (1 => 'apples', 2 => 'bananas', 3 => 'oranges');
```

因为PHP在处理脚本中的空白时非常自由，所以可以使用下面这种多行的样式让结构更加易于阅读：

```
$list = array (
1 => 'apples',
2 => 'bananas',
3 => 'oranges'
);
```

最后，指定的索引值并不必须为数值，还可以使用单词。这种索引的技术在制作更加有意义的列表时非常实用。作为示例，可以利用下面的脚本创建一个数组来记录一周中每天供应的汤。该示例也将诠释如何能够以及不能打印数组（虽然已经讲过，但需要反复练习）。

⇒ 创建一个数组

(1) 在文本编辑器或者IDE中创建一个新的文档，命名为soups1.php（参看脚本7-1）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type"content="text/html; charset=utf-8"/>
  <title>No Soup for You!</title>
</head>
<body>
<h1>Mmm...soups</h1>
```

脚本7-1　$soups数组包含3个元素并且使用字符串作为它的键

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>No Soup for You!</title>
7   </head>
8   <body>
9   <h1>Mmm...soups</h1>
10  <?php // 脚本7-1 - soups1.php
11  /* 脚本创建并打印出一个数组。*/
```

```
12 // 可以在此进行错误处理。
13
14 // 创建数组:
15 $soups = array (
16 'Monday' => 'Clam Chowder',
17 'Tuesday' => 'White Chicken Chili',
18 'Wednesday' => 'Vegetarian'
19 );
20
21 // 尝试打印这个数组:
22 print "<p>$soups</p>";
23
24 // 打印数组的内容:
25 print_r ($soups);
26
27 ?>
28 </body>
29 </html>
```

(2) 开始脚本中的PHP片段，如果需要，添加错误处理：

```
<?php // 脚本7-1 - soups1.php
```

如果没有开启display_errors，或者error_reporting被设置为错误的级别，请参见第3章，在这里添加适当的代码以改变这些设置。

(3) 使用array()函数创建一个数组：

```
$soups = array (
'Monday' => 'Clam Chowder',
'Tuesday' => 'White Chicken Chili',
'Wednesday' => 'Vegetarian'
);
```

这是在PHP中用字符串作为索引对数组进行初始化（创建和为其赋值）所使用的正确格式。由于其键和值都是字符串类型，因此需要用单引号将之引用。

不要在最后一个数组元素之后不经意地添加逗号（索引为*Wednesday*），这样做会导致解析错误发生。

(4) 尝试打印这个数组：

```
print "<p>$soups</p>";
```

正如即将看到的那样，数组与其他变量的不同之处还表现为它们不能用打印其他（标量）变量的方式进行打印。

(5) 使用与前述不同的print_r()函数打印数组：

```
print_r ($soups);
```

在第2章中我们已经学习了如何使用print_r()函数来显示任何变量的内容和结构。在这里使用它，可以看到这个函数的工作方式与print处理数组方式的不同之处。

(6) 结束PHP和HTML代码部分：

```
?>
```

```
</body>
</html>
```

(7) 将文档保存为soups1.php，放置在启用了PHP的服务器上适当的目录中，并且在Web浏览器中进行测试（参见图7-1）。

记住，要通过URL来运行PHP脚本。

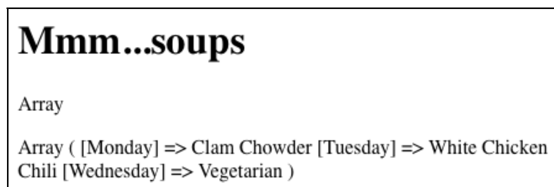


图7-1　由于数组和其他变量类型是结构上不同的，因此打印数组的请求会导致单词“数组”的出现（以及错误的出现，这取决于PHP的版本及其设置）。另外一方面，print_r()函数用来打印数组的内容和结构

✓提示

- 索引从0开始是PHP以及大多数编程语言的标准惯例。这种计数系统看起来如此不自然，但是却需要将之坚持下去。可以用下面两种技术来应对：第一种，手动将数组中所有的索引起始都指定为1；第二种，忘记从1开始计数的方式。可以尝试用哪个方式更加简便，但是大多数程序员都已经适应了这种奇怪的结构。
- 必须在引用一个数组的元素时使用与创建数组时使用的相同的索引。在$soups示例中，即使该数组显然拥有第一个元素（第一个元素通常被索引为数字0），但是$soups[0]并没有值。
- 如果使用array()函数来定义索引，那么可以指定第一个元素的索引，其他元素的索引将延续第一个元素的索引值。例如：

```
$list = array (1 => 'apples', 'bananas', 'oranges');
```

  现在bananas的索引为2而oranges的索引为3。
- range()函数也可以用来创建数组，同时将数组中元素的值限定于一定范围。下面是这样的两个示例：

```
$ten = range (1, 10);
$alphabet = range ('a', 'z');
```

- 在PHP第5版中，range()函数包含一个可以用来指定增量的参数：

```
$evens = range (0, 100, 2);
```

- 如果在脚本中用var_dump()函数代替print_r()，它将不仅显示数组的内容，同时还会以更加详细的格式呈现数组的结构（参见图7-2）。
- 键为数值的数组被称为*索引*（index）数组。如果键为字符串，则被称为*关联*（associative）数组。在其他语言中则称关联数组为*散列表*（hash）。

```
array(3) { ["Monday"]=> string(12) "Clam Chowder" ["Tuesday"]=>
string(19) "White Chicken Chili" ["Wednesday"]=> string(10)
"Vegetarian" }
```

图7-2　var_dump()函数（在脚本7-1中用来替代print_r()函数）显示了在数组中的元素数量以及每个字符串值的长度

7.3　向数组添加项

在PHP中创建了一个数组之后，就可以使用赋值运算符（等号）向这个数组添加额外的元素，这个方法同为字符串或者数值变量进行赋值非常类似。当进行这样的操作时，既可以为所添加的元素指定键也可以不指定键，但是在其中任意一种情况下，都必须将数组用方括号进行引用。可以编写下面的代码向$list数组添加两个项：

```
$list[ ] = 'pears';
$list[ ] = 'tomatoes';
```

如果向这个已存在的数组添加元素而没有指定键，那么每个元素的索引都将延续已有索引的数值。假设这个数组同前述章节中的数组一样，已存在索引1、2、3，那么pears现在的索引将为4，而tomatoes则为5。

如果确实指定了索引，那么它的值将被指定为相应的位置。任何已存在的值此时都将被覆盖，就像这样：

```
$list[3] = 'pears';
$list[4] = 'tomatoes';
```

现在，在数组中第四个元素的值为tomatoes，并且$list中没有元素的值为oranges（其值被pears覆盖了）。出于这一点的考虑，除非打算要覆盖某些已经存在的数据，最好不要在向数组添加值的时候为特定的键命名。

但是，如果数组使用字符串作为索引，很可能要指定键，以确保不出现字符串和数值键的奇怪组合。

测试这个过程，我们将重写soups1.php以为数组添加更多的元素。这里还将会在添加新元素之前和之后打印数组中的元素数量，以查看添加更多元素造成的区别。

使用strlen()可以获取一个字符串所包含的字符数量，从而了解获知该字符串的长度。类似地，可以通过使用count()以了解在数组中的元素数量：

```
$how_many = count($array);
```


删除数组和数组元素

我们一般不需要频繁地从数组中删除独立的元素，但是可以用unset()函数来完成这个功能。这个函数能够用来删除变量以及释放它所占用的内存。当为一个数组元素应用这个函数时，该元素就会被删除：

```
unset($array[4]);
unset($array['name']);
```

如果为整个数组或者任何其他的变量类型使用unset()函数，那么整个变量都将被删除：

```
unset($array);
unset($string);
```

还可以使用array()函数来重置（reset）数组（将数组的值清空而不删除数组）：

```
$array = array();
```

这样做具有变量初始化的效果：让数组依旧存在并且在不赋值的情况下定义它的类型。

⇛ 向一个数组添加元素

(1) 如果soups1.php不在开启状态的话，在文本编辑器或IDE中打开它。

(2) 在用array()对数组进行初始化之后，添加下面的脚本（参看脚本7-2，命名为soups2.php）：

```
$count1 = count ($soups);
print "<p>The soups array originally had $count1 elements.</p>";
```

脚本7-2　可以通过使用赋值运算符为每个元素赋值的方式直接向数组添加元素。count()函数将帮助持续跟踪数组中包含的元素数量

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>No Soup for You!</title>
7   </head>
8   <body>
9   <h1>Mmm...soups</h1>
10  <?php // 脚本7-2 - soups2.php
11  /* 脚本创建并打印出一个数组。*/
12  // 可以在此进行错误处理。
13
14  // 创建数组:
15  $soups = array (
16  'Monday' => 'Clam Chowder',
17  'Tuesday' => 'White Chicken Chili',
18  'Wednesday' => 'Vegetarian'
19  );
20
21  // 计算并打印数组中当前元素个数:
22  $count1 = count ($soups);
23  print "<p>The soups array originally had $count1 elements.</p>";
24
25  // 向数组中添加3个元素:
26  $soups['Thursday'] = 'Chicken Noodle';
27  $soups['Friday'] = 'Tomato';
```

```
28  $soups['Saturday'] = 'Cream of Broccoli';
29
30  // 再次计算并打印数组中元数个数:
31  $count2 = count ($soups);
32  print "<p>After adding 3 more soups, the array now has $count2 elements.</p>";
33
34  // 打印数组内容:
35  print_r ($soups);
36
37  ?>
38  </body>
39  </html>
```

count()函数用来获知在$soups中的元素数量。将这个值赋给一个变量值，可以很容易地将结果打印出来。

(3) 为数组再添加3个元素：

```
$soups['Thursday'] = 'Chicken Noodle';
$soups['Friday'] = 'Tomato';
$soups['Saturday'] = 'Cream of Broccoli';
```

这些代码又向已有的数组添加了3种汤，分别用Thursday、Friday和Saturday进行索引。

(4) 重新对数组中的元素数量进行计数，并且将值打印出来。

```
$count2 = count ($soups);
print "<p>After adding 3 more soups, the array now has $count2 elements.</p>";
```

第二个print调用是对第一个的重复，为了了解数组现在所包含的元素数量。

(5) 删除：

```
print"<p>$soups</p>";
```

我们已经不再需要该行代码了，因此可以将它删除（现在知道不能这样简单地打印数组）。

(6) 将脚本保存为soups2.php，放置在启用了PHP的服务器上适当的目录中，并在Web浏览器中进行测试（参见图7-3）。

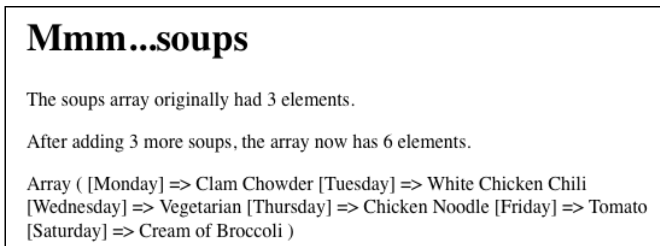


图7-3　确定向数组成功添加元素的直接方法是在添加操作的前后对元素的数量进行计数比较

✓提示

❑ 当直接向数组添加元素时需要非常小心。正确的处理方法为：

```
$array[ ] = 'Add This';
```

或者

```
$array[1] = 'Add This';
```

以及错误的方法：

```
$array = 'Add This';
```

如果忘记了使用方括号，那么添加的值将替代整个已有的数组，只留下一个字符串或者数值。

- 代码（`$array[ ] ='Value'`）；会创建变量`$array`，如果它尚不存在的话。
- 在使用这些数组时，本书用单引号来引用键和值。没有必要进行插值（就像一个变量那样），因此双引号在这里并不是必须的。但是如果愿意，也可以在这里使用双引号。
- 如果键都是数值、变量或者常数（本书将在第8章中涉及），那么不用（并且事实上不应该）将之用引号引用。例如：

```
$day = 'Sunday';
$soups[$day] = 'Mushroom';
```

- `sizeof()`函数是`count()`的别名。它也返回在数组中的元素数量。

数组合并

PHP中有一个函数可以用来把一个数组附加到其他的数组上。这可以认为是数组的串联。这个函数名为`array_merge()`，可以用下面的方式进行使用：

```
$new_array = array_merge($array1, $array2);
```

你可以使用这个函数编写`soups2.php`页：

```
$soups2 = array (
'Thursday' => 'Chicken Noodle',
'Friday' => 'Tomato',
'Saturday' => 'Cream of Broccoli',
);
$soups = array_merge($soups, $soups2);
```

甚至可以用加号（将两个数组加在一起）实现这个结果：

```
$soups = $soups + $soups2;
```

或者

```
$soups += $soups2;
```

7.4　访问数组元素

无论是如何创建数组的，从数组中检索某个特定元素（或者值）只有一个方法，那就是用它

的索引。

```
print "The first item is $array[0]";
```

如果数组的索引使用字符串，则需要用引号来引用，那么必须调整索引使用的引号，这是因为它们会同print语法相冲突。下面这行代码将引发这样的问题（参见图7-4）：

```
print "<p>Monday's soup is $soups['Monday'].</p>";
```

为了解决这个问题，我们可以用花括号将整个数组结构括起来（参见图7-5）：

```
print "<p>Monday's soup is {$soups['Monday']}.</p>";
```

具有讽刺意味的是，数组能够在一个变量中存贮多个值的特性让它很有用，但是同时又带来其他变量类型所没有的限制：必须知道数组的键才能够访问它的元素。如果该数组被设定为使用字符串作为键（如数组$soups），那么$soups[1]则不指向任何值（参见图7-6）。出于这个原因，同时变量不区分大小写，$soups['monday']变得毫无意义，这是因为Clam Chowder是被索引指向$soups['Monday']的。

Parse error: syntax error, unexpected T_ENCAPSED_AND_WHITESPACE, expecting T_STRING or T_VARIABLE or T_NUM_STRING in **/Users/larryullman/Sites/phpvqs4/soups2.php** on line **37**

图7-4　在关联数组中用双引号引用特定的元素会导致解析错误

Monday's soup is Clam Chowder.

图7-5　用花括号封装数组元素引用是解决解析错误的方法之一

Notice: Undefined offset: 1 in **/Users/larryullman/Sites/phpvqs4/soups2.php** on line **37**

图7-6　引用并不存在的数组索引会导致出现Undefined offset或Undefined Index 通知信息

访问数组中所有值的最快捷而且最简单的方法是使用foreach循环。这个循环结构将遍历数组中的每个元素：

```
foreach ($array as $key => $value) {
        print "<p>Key is $key. Value is $value</p>";
}
```

循环的每次迭代，都会将当前数组元素的键赋值给$key变量，将元素的值赋值给$value变量。注意，这里可以使用任意变量：$k和$v都可以。

现在我们可以利用这些知识编写一个新的包含各种汤的脚本。

⇒ 打印数组中的值

(1) 在文本编辑器或者IDE中新建一个文档，命名为soups3.php（参看脚本7-3）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>No Soup for You!</title>
</head>
<body>
<h1>Mmm...soups</h1>
```

脚本7-3　foreach循环是访问数组中每个元素的最简便的方法

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>No Soup for You!</title>
7   </head>
8   <body>
9   <h1>Mmm...soups</h1>
10  <?php // 脚本7-3 - soups3.php
11  /* 脚本创建并打印出一个数组。*/
12
13  // 可以在此进行错误处理。
14
15  // 创建数组:
16  $soups = array (
17  'Monday' => 'Clam Chowder',
18  'Tuesday' => 'White Chicken Chili',
19  'Wednesday' => 'Vegetarian',
20  'Thursday' => 'Chicken Noodle',
21  'Friday' => 'Tomato',
22  'Saturday' => 'Cream of Broccoli'
23  );
24
25  // 打印数组中的键和值:
26  foreach ($soups as $day =  > $soup) {
27    print "<p>$day: $soup</p>\n";
28  }
29
30  ?>
31  </body>
32  </html>
```

(2) 开启页面中的PHP代码片段，如果愿意的话，添加错误管理：

```
<?php // 脚本7-3 - soups3.php
```

(3) 创建$soups数组：

```
$soups = array (
'Monday' => 'Clam Chowder',
'Tuesday' => 'White Chicken Chili',
'Wednesday' => 'Vegetarian',
'Thursday' => 'Chicken Noodle',
'Friday' => 'Tomato',
```

```
'Saturday' => 'Cream of Broccoli'
);
```

在这里一次就创建了整个数组，虽然可以使用同前面脚本中相同的方式（逐步创建数组）。

(4) 创建一个foreach循环以打印每天提供的汤：

```
foreach ($soups as $day => $soup)
{
  print "<p>$day: $soup</p>\n";
}
```

foreach循环遍历数组$soups的每个元素，将每个索引赋值给$day，并将每个值赋值给$soup。这些值现在打印在HTML段落标记中。该print语句包含一个换行符（使用\n创建的），它将对页面中的HTML源代码造成影响。

(5) 结束PHP部分和HTML页面：

```
?>
</body>
</html>
```

(6) 将页面保存为soups3.php，并放置在启用了PHP的服务器上适当的目录中，然后在Web浏览器中进行测试（参见图7-7）。

Mmm...soups

Monday: Clam Chowder

Tuesday: White Chicken Chili

Wednesday: Vegetarian

Thursday: Chicken Noodle

Friday: Tomato

Saturday: Cream of Broccoli

图7-7 为数组中的每个元素执行循环后将生成这个页面。foreach结构允许脚本对每个键和值进行访问而无需事先对它们是什么进行了解

7

✓提示

- 使用数组的方式之一就是用赋值运算符将特定元素的值赋给单独的变量：

  ```
  $total = $array[1];
  ```

  这样做可以在保留数组原始值的同时仍然能够将它作为独立的变量来操作。

- 如果只需要访问一个数组的值（而不是它的键），那么可以使用foreach结构：

  ```
  foreach ($array as $value) {
    // 做任何想做的事情。
  }
  ```

- 访问数组中所有元素的另外一个方法是使用for循环：

  ```
  for ($n = 0; $n < count($array); $n++) {
    print "The value is $array[$n]";
  }
  ```

- 在打印用字符串作为键的数组的值时，请记住用花括号来避免错误发生。这里的示例使用的引号并没有问题，因此不需要花括号：

  ```
  $name = trim ($array['name']);
  $total = $_POST['qty'] *
  $_POST['price'];
  ```

- 花括号也可以用来分开变量起始符和美元符号，或其他字符：

  ```
  print "The total is ${$total}.";
  ```

7.5　创建多维数组

多维数组表现得既简单又复杂。它的结构和概念有一些难以掌握，但是用PHP来创建和访问多维数组却相当简单。

多维数组可以用来包含比标准数组更多的信息，而这是通过用其他数组作为它的值而不是简单的字符串和数值作为值来实现的。例如：

```
$fruits = array ('apples', 'bananas', 'oranges');
$meats = array ('steaks', 'hamburgers', 'pork chops');
$groceries = array (
'fruits' => $fruits,
'meats' => $meats,
'other' => 'peanuts',
'cash' => 30.00
);
```

数组`$groceries`现在由一个字符串（`peanuts`）、一个浮点数值（`30.00`）和2个数组（`$fruits`和`$meats`）组成。

指向多维数组中的某个元素会有一点复杂。要点是根据需要在方括号中继续添加索引。因此在这个示例中，`bananas`位于`$groceries['fruits'][1]`。

首先，使用`['fruits']`来指向数组`$groceries`中的元素（在这种情况下，是一个数组）。然后再基于它的位置指向数组中的这个元素：它是第二项，因此需要使用索引`[1]`。

在下一个任务中，我们将编写一个脚本来创建另外一个多维数组的示例。

⇛ 使用多维数组

(1) 在文本编辑器或者IDE中开启一个新的文档，命名为`books.php`（参看脚本7-4）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html;
  →charset=utf-8"/>
  <title>Larry Ullman's Books and Chapters</title>
</head>
<body>
<h1>Some of Larry Ullman's Books</h1>
```

脚本7-4　多维数组`$books`在一个大变量中存储了大量的信息

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html;
      charset=utf-8"/>
6     <title>Larry Ullman's Books and Chapters</title>
7   </head>
```

```
8   <body>
9   <h1>Some of Larry Ullman's Books</h1>
10  <?php // 脚本7-4 - books.php
11  /* 脚本创建并打印出一个多维数组。*/
12  // 可以在此进行错误处理。
13
14  // 创建第1个数组:
15  $phpvqs = array (1 => 'Getting Started with PHP', 'Variables', 'HTML Forms
    and PHP', 'Using Numbers');
16
17  // 创建第2个数组:
18  $phpadv = array (1 => 'Advanced PHP Techniques', 'Developing Web
    Applications', 'Advanced Database Concepts', 'Security Techniques');
19
20  // 创建第3个数组:
21  $phpmysql = array (1 => 'Introduction to PHP', 'Programming with PHP',
    'Creating Dynamic Web Sites', 'Introduction to MySQL');
22
23  // 创建多维数组:
24  $books = array (
25  'PHP VQS' => $phpvqs,
26  'PHP Advanced VQP' => $phpadv,
27  'PHP and MySQL VQP' => $phpmysql
28  );
29
30  // 打印出一些值:
31  print "<p>The third chapter of my first book is <i>{$books['PHP
    VQS'][3]}</i>.</p>";
32  print "<p>The first chapter of my second book is <i>{$books['PHP
    Advanced VQP'][1]}</i>.</p>";
33  print "<p>The fourth chapter of my fourth book is <i>{$books['PHP
    and MySQL VQP'][4]}</i>.</p>";
34
35  // 使用foreach遍历数组:
36  foreach ($books as $key => $value) {
37    print "<p>$key: $value</p>\n";
38  }
39
40  ?>
41  </body>
42  </html>
```

(2) 创建初始PHP标签，如果需要请添加错误管理：

```
<?php // 脚本7-4 - books.php
```

(3) 创建第一个数组：

```
$phpvqs = array (1 => 'Getting Started', 'Variables', 'HTML Forms
→and PHP', 'Using Numbers');
```

构建这个多维数组，我们先创建3个标准数组，然后将它们作为这个更大数组的值。该数组

（被称为$phpvqs，是*PHP for the World Wide Web: Visual QuickStart Guide*的简称）使用数值作为键，同时使用字符串作为值。数值从1开始对应章节编号。值则为每章的标题。

(4) 创建接下来的两个数组：

```
$phpadv = array (1 => 'Advanced PHP Techniques', 'Developing Web
→Applications', 'Advanced Database Concepts', 'Security Techniques');
$phpmysql = array (1 => 'Introduction to PHP', 'Programming with PHP',
→'Creating Dynamic Web Sites', 'Introduction to MySQL');
```

对于每个数组，出于简便的考虑，只添加本书前4章的信息。

(5) 创建主体，多维数组：

```
$books = array (
'PHP VQS' => $phpvqs,
'PHP 5 Advanced VQP' => $phpadv,
'PHP 6 and MySQL 5 VQP' => $phpmysql
);
```

数组$books是本脚本中的主数组。它使用字符串作为键（本书标题的缩写版）并且数组作为值。可以用函数array()像创建其他数组那样创建它。

(6) 打印本书第3章的章名：

```
print "<p>The third chapter of my first book is <i>{$books['PHP
→VQS'][3]}</i>.</p>";
```

按照之前介绍的规则，为了访问所有单独章的章名，我们所需要做的就是从$books开始，首先是第一个索引（['PHP VQS']），然后是下一个索引（[3]）。

因为我们将把这些内容放置在一个print调用中，所以需要用花括号将整个结构引用，以避免发生解析错误。

(7) 再打印两个示例：

```
print "<p>The first chapter of my second book is <i>{$books
→['PHP Advanced VQP'][1]}</i>.</p>";
print "<p>The fourth chapter of my fourth book is <i>{$books
→['PHP and MySQL VQP'][4]}</i>.</p>";
```

(8) 通过foreach循环来遍历数组$books以查看结果：

```
foreach ($books as $key => $value) {
  print "<p>$key: $value</p>\n";
}
```

变量$key存储了书中每个标题的缩写，并且$value变量包含每个章节数组。

(9) 结束PHP代码片段并且完成HTML页面：

```
?>
</body>
</html>
```

(10) 将文件保存为books.php，并保存在启用了PHP服务器上适当的目录中，然后在Web浏览器中进行测试（参见图7-8）。

Some of Larry Ullman's Books

The third chapter of my first book is *HTML Forms and PHP*.

The first chapter of my second book is *Advanced PHP Techniques*.

The fourth chapter of my fourth book is *Introduction to MySQL*.

PHP VQS: Array

PHP Advanced VQP: Array

PHP and MySQL VQP: Array

图7-8　开始的3行由print语句生成。最后3行显示了foreach循环的结果（以及因为试图打印数组而产生的提示）

✓提示

- 为了访问每个数组中的每个元素，可以将两个foreach循环嵌套使用，如下（参见图7-9）：

```
foreach ($books as $title => $chapters) {
  print "<p>$title";
  foreach ($chapters as $number => $chapter) {
    print "<br/>Chapter $number is $chapter";
  }
  print '</p>';
}
```

- 使用函数print_r()或者函数var_dump()（将函数放置在HTML中的<pre>标签范围内可以获得更好的显示格式）可以对整个多维数组进行查看（参见图7-10）。

PHP VQS
Chapter 1 is Getting Started with PHP
Chapter 2 is Variables
Chapter 3 is HTML Forms and PHP
Chapter 4 is Using Numbers

PHP Advanced VQP
Chapter 1 is Advanced PHP Techniques
Chapter 2 is Developing Web Applications
Chapter 3 is Advanced Database Concepts
Chapter 4 is Security Techniques

PHP and MySQL VQP
Chapter 1 is Introduction to PHP
Chapter 2 is Programming with PHP
Chapter 3 is Creating Dynamic Web Sites
Chapter 4 is Introduction to MySQL

图7-9　一个foreach循环包含在另外一个foreach循环中，能够实现对一个二维数组中每个元素的访问

```
Array
(
    [PHP VQS] => Array
        (
            [1] => Getting Started with PHP
            [2] => Variables
            [3] => HTML Forms and PHP
            [4] => Using Numbers
        )

    [PHP Advanced VQP] => Array
        (
            [1] => Advanced PHP Techniques
            [2] => Developing Web Applications
            [3] => Advanced Database Concepts
            [4] => Security Techniques
        )

    [PHP and MySQL VQP] => Array
        (
            [1] => Introduction to PHP
            [2] => Programming with PHP
            [3] => Creating Dynamic Web Sites
            [4] => Introduction to MySQL
        )

)
```

图7-10　函数var_dump()显示了数组$books的结构和内容

- 可以使用一系列嵌套的`array()`调用的方式在一个语句中创建多维数组（用以代替本示例中使用多个步骤的方式）。但是，并不推荐这样做，这是因为随着语句中嵌套数量的增加，引发语法错误的可能性就越大。
- 尽管本示例中的所有子数组都有相同的结构（用数值作为索引，并且有4个元素），但这并不是多维数组所必须的。
- 为了了解关于“Larry Ullman Collection”的更多信息，以及这里涉及的三本书，请访问本书的站点：www.LarryUllman.com。

7.6　数组排序

PHP提供多种排序数组的方法（如果被排序的值是字符串，可以使用字母顺序排序法；如果被排序的值是数值，则可以按照数值方式进行排序）。当排序一个数组时，必须牢记数组是由键—值对组成的。因此，一个数组能够基于键或者值来进行排序。在依照值进行排序的同时保持相应的键与之对应是非常复杂的，或者也可以对值进行排序，然后再为它们赋予新的键。

可以使用`sort()`来对值进行排序而不用考虑键的问题，而使用`rsort()`可以对这些值进行反向排序（同样不用考虑键）。每种排序函数的语法如下：

```
function($array);
```

因此，`sort()`和`rsort()`的用法如下：

```
sort($array);
rsort($array);
```

可以使用`asort()`来实现在对值进行排序的同时保持每个值同它键之间的对应关系，而使用`arsort()`来实现在对值进行反向排序的同时保持每个值同它键之间的对应关系。

可以使用`ksort()`来实现在对键进行排序的同时保持每个键同它值之间的对应关系，而使用`krsort()`来实现在对键进行反向排序的同时保持每个键同它值之间的对应关系。表7-2列举了这些函数。

表7-2　数组排序函数

函　　数	排序依据	是否保持键–值对应关系
sort()	Values	No
rsort()	Values (inverse)	No
asort()	Values	Yes
arsort()	Values (inverse)	Yes
ksort()	Keys	Yes
krsort()	Keys (inverse)	Yes

最后，`shuffle()`用来对数组中的元素顺序进行随机重组。

下面是数组排序的示例，这里将创建一个列表，其中包含学生的名称以及他们在测验中获得的成绩，然后按照成绩和姓名来排序这个列表。

⇛ 对数组进行排序

(1) 在文本编辑器或者IDE中新建一个文档，命名为sort.php（参看脚本7-5）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
  <title>My Little Gradebook
</title>
</head>
<body>
```

脚本7-5 PHP为数组排序提供多种不同的函数，其中包括（这里使用的）arsort()和ksort()

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>My Little Gradebook</title>
7   </head>
8   <body>
9   <?php // 脚本7-5 - sort.php
10  /* 脚本创建、排序并打印出一个数组。*/
11
12  // 可以在此进行错误处理。
13
14  // 创建数组:
15  $grades = array(
16  'Richard' => 95,
17  'Sherwood' => 82,
18  'Toni' => 98,
19  'Franz' => 87,
20  'Melissa' => 75,
21  'Roddy' => 85
22  );
23
24  // 以原始顺序打印数组:
25  print '<p>Originally the array looks like this: <br/>';
26  foreach ($grades as $student => $grade) {
27    print "$student: $grade<br/>\n";
28  }
29  print '</p>';
30
31  // 倒序数组并打印。
32  arsort ($grades);
33  print '<p>After sorting the array by value using arsort(), the array looks
    like this: <br/>';
34  foreach ($grades as $student => $grade) {
35    print "$student: $grade<br/>\n";
36  }
```

```
37  print '</p>';
38
39  // 以键排序数组并打印。
40  ksort ($grades);
41  print '<p>After sorting the array by key using ksort(), the array looks like
    this: <br/>';
42  foreach ($grades as $student => $grade) {
43    print "$student: $grade<br/>\n";
44  }
45  print '</p>';
46
47  ?>
48  </body>
49  </html>
```

(2) 开始PHP代码部分，如果需要请添加错误管理：

```
<?php // 脚本7-5 - sort.php
```

(3) 创建数组：

```
$grades = array(
'Richard' => 95,
'Sherwood' => 82,
'Toni' => 98,
'Franz' => 87,
'Melissa' => 75,
'Roddy' => 85
);
```

数组`$grades`由6个学生的姓名以及他们相应的成绩组成。因为成绩是数值类型的值，因此它们不需要在赋值时被引号引用。

(4) 打印一个标题，然后用`foreach`循环打印出数组中的每个元素：

```
print '<p>Originally the array looks like this: <br/>';
foreach ($grades as $student => $grade) {
  print "$student: $grade<br/>\n";
}
print '</p>';
```

`$grades`数组会打印三次，指示数组状态的标题行对我们来说很有用。首先，脚本按照原始顺序打印数组。可以用一个`foreach`循环来做到，此时每个索引（学生的姓名）都被赋值给`$student`，并且每个值（学生的成绩）都被赋值给`$grade`。最后的`print`调用将关闭HTML段落。

(5) 将数组基于值进行倒序排列以查看谁获得了最高的成绩：

```
arsort ($grades);
```

因为要了解谁获得了最高的成绩，需要使用`arsort()`而不是`asort()`。后者按照数值升序排序数组，将会对评分进行如下排列：75、82、85等，而不是按照期望的98、95、87方式。

还必须使用`arsort()`而不是`rsort()`，用于维护键—值关系（`rsort()`会忽略与评级相关联的学生名字）。

(6) 用另外一个循环再次打印数组（带有标题）：

```
print '<p>After sorting the array by value using arsort(), the array
→looks like this: <br/>';
foreach ($grades as $student => $grade) {
  print "$student: $grade<br/>\n";
}
print '</p>';
```

(7) 基于键对数组进行排序以让数组中学生的姓名依照字母顺序进行排列：

```
ksort ($grades);
```

函数`ksort()`对数组依照键进行了重组（在当前情况下为字母顺序）的同时保持了键–值的对应关系。

(8) 最后一次打印标题和数组：

```
print '<p>After sorting the array by key using ksort(), the array
→looks like this: <br/>';
foreach ($grades as $student => $grade) {
  print "$student: $grade<br/>\n";
}
print '</p>';
```

(9) 用标准的PHP和HTML标签完成脚本：

```
?>
</body>
</html>
```


(10) 将脚本保存为`sort.php`，放置在启用了PHP服务器上适当的目录中，并在Web浏览器上进行测试（参见图7-11）。

Originally the array looks like this:
Richard: 95
Sherwood: 82
Toni: 98
Franz: 87
Melissa: 75
Roddy: 85

After sorting the array by value using arsort(), the array looks like this:
Toni: 98
Richard: 95
Franz: 87
Roddy: 85
Sherwood: 82
Melissa: 75

After sorting the array by key using ksort(), the array looks like this:
Franz: 87
Melissa: 75
Richard: 95
Roddy: 85
Sherwood: 82
Toni: 98

图7-11　可以对数组进行各种排序得到不同的结果。请在选择排序函数时密切留意对是否希望保持键–值的对应关系

✓提示

- 数组$grades可以创建为用成绩作为键、学生姓名作为值的方式。两种方式都可行。
- 函数natsort()和natcasesort()排序字符串（同时保持键—值对应关系）时使用*自然顺序*（natural order）的方式。自然顺序排列方式最显而易见的示例就是它会将name2排在name12之前，反之sort()处理的结果是name12在name2之前。
- 函数usort()、uasort()和ursort()可以对数组使用用户自定义的比较函数方式进行排序。这些函数通常都应用在多维数组中。

7.7　字符串和数组之间的转换

现在我们已经对字符串和数组都有所了解，本节将介绍在这两种形式之间进行转换的两个函数。第一个是implode()，它用来将数组转换为字符串，第二个是explode()用来做相反的处理。

以下是使用这些函数的原因。

- 将数组转换为字符串是为了能够将值附加在URL上进行传递（而不能用数组如此轻松地实现）。
- 将数组转换为字符串是为了将信息储存在数据库中。
- 将字符串转换为数组是为了将一个用逗号分隔的文本区（例如表单中的关键字搜索区域）转化为相互独立的形式。

使用explode()的语法如下：

```
$array = explode($separator, $string);
```

*separator*分隔符指明了一个或多个字符，用来区分一个值的结束和另外一个值开始。通常情况下分隔符是一个逗号、一个制表符或者一个空格。因此代码可能如下所示：

```
$array = explode(',', $string);
```

或

```
$array = explode(' ', $string);
```

为了将一个数组转换为字符串，需要定义用什么作为分隔符（也就是之间的连接符），PHP将处理剩下的工作：

```
$string = implode($glue, $array);
$string = implode(',', $array);
```

或

```
$string = implode(' ', $array);
```

下面诠释如何使用explode()和implode()，这里将创建一个HTML表单用以包含一个以空格进行分隔的代表用户姓名的字符串（参见图7-12）。PHP脚本将会把字符串转换为数组以便能够对列表进行排序。最后，代码将创建并返回被按照字母顺序排列的字符串（参见图7-13）。

Enter the words you want alphabetized with each individual word separated by a space:

Brian Sommar Eric Mark Shauna Allison Mike [Alphabetize!]

图7-12 这个HTML表单有一个词语列表，它被list.php脚本按照字母顺序进行排序

An alphabetized version of your list is:
Allison
Brian
Eric
Mark
Mike
Shauna
Sommar

图7-13 这是一个相同的列表，按照字母顺序对用户进行排序。处理过程很迅速并且很容易编写代码，但是如果没有数组将无法这样轻松实现

⇛ 创建HTML表单

(1) 在文本编辑器或者IDE中开启一个新的文档，命名为list.html（参看脚本7-6）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>I Must Sort This Out! </title>
</head>
<body>
<!-- 脚本7-6 - list.html -->
```

脚本7-6 这是一个简单的HTML表单，用户可以通过它来提交一个词语列表。其中还包含对应当如何在谨慎的Web设计策略中使用表单进行了详细的介绍

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>I Must Sort This Out!</title>
7   </head>
8   <body>
9   <!-- 脚本7-6 - list.html -->
10  <div><p>Enter the words you want alphabetized with each individual word
    separated by a space:</p>
11
12  <form action="list.php" method="post">
13
14    <input type="text" name="words" size="60" />
15    <input type="submit" name="submit" value="Alphabetize!" />
16
```

```
17  </form>
18  </div>
19  </body>
20  </html>
```

(2) 创建一个有文本输入框的HTML表单：

```
<div><p>Enter the words you want alphabetized with each individual
→word separated by a space:</p>
<form action="list.php" method="post">
  <input type="text" name="words" size="60" />
```

在这种情况下对用户做出提示是很重要的。例如，如果他们输入一个逗号分隔的列表，这样并不能正确地处理这个字符串（在完成了两段脚本之后，尝试使用逗号而不是空格进行分隔，看看会发生什么）。

(3) 创建一个提交按钮，并且结束表单和HTML页面：

```
  <input type="submit" name="submit" value="Alphabetize!" />
</form>
</div>
</body>
</html>
```

(4) 将脚本保存为list.html，并放置在启用了PHP服务器上适当的目录中。

现在我们将编写list.php页面来处理由list.html生成的数据。

⇛ 在字符串和数组中进行转换

(1) 在文本编辑器或者IDE中开启一个新的文档，命名为list.php（参看脚本7-7）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>I Have This Sorted Out </title>
</head>
<body>
<?php // 脚本7-7 - list.php
```

脚本7-7　因为函数explode()和implode()是如此简单并且强大，你可以快速并容易地通过编写几行代码来对提交的（几乎任何长度）词语列表进行排序

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>I Have This Sorted Out</title>
7   </head>
8   <body>
9   <?php // 脚本7-7 - list.php
10  /* 脚本在$_POST['words']接收一个字符串。
```

```
11  接下来将字符串转换为数组，按字母顺序排序数组并打印出来。*/
12
13  // 可以在此进行错误处理。
14
15  // 将引入的字符串转换为数组：
16  $words_array = explode(' ' , $_POST ['words']);
17
18  // 排序数组：
19  sort($words_array);
20
21  // 将数组转换为字符串：
22  $string_words = implode('<br/>', $words_array);
23
24  // 打印结果：
25  print "<p>An alphabetized version of your list is: <br/>$string_words</p>";
26
27  ?>
28  </body>
29  </html>
```

(2) 将引入的字符串`$_POST['words']`转换为数组：

```
$words_array = explode(' ' , $_POST['words']);
```

该行代码依照字符串`$_POST['words']`创建了一个新的数组：`$words_array`。在`$_POST['words']`中词语和词语之间的空格指明下一个词语为一个新的数组元素。因此第一个词语为`$words_array[0]`，然后在`$_POST['words']`中有一个空格，接着第二个词语为`$words_array[1]`，依此类推，直到`$_POST['words']`的结尾。

(3) 对数组按照字母顺序进行排序：

```
sort($words_array);
```

因为不需要保持`$words_array`中的键–值对应关系，因此这里可以用`sort()`替代之前使用的`asort()`。

(4) 依照排序后的数组创建新的字符串：

```
$string_words = implode('<br/>', $words_array);
```

数组不能像字符串那样很容易被打印处理，因此需要将数组`$words_array`转换为字符串`$string_words`。结果字符串以`$words_array[0]`的值开头，接下来为HTML`<br/>`标签、`$words_array[1]`的值，依此类推。使用`<br/>`来替代空格或者逗号将会使在浏览器中打印出来的格式更易于阅读。

(5) 在浏览器中打印新的字符串：

```
print "<p>An alphabetized version of your list is: <br/> $string_words</p>";
```

(6) 结束PHP代码片段和HTML页面：

```
?>
</body>
</html>
```

7

(7) 将页面保存为`list.php`，放置在和`list.html`相同的目录中，并且在Web浏览器中对两个脚本进行测试（参见图7-12和图7-13）。

✓提示

- ❑ 也可以运行用`join()`函数编写的代码，它同`implode()`是同义的。

7.8　在表单中创建数组

通过本章的学习，我们已经在PHP页面中完整地创建了数组。然而，可以通过一个HTML表单向PHP脚本发送数组的数据。事实上，每当使用`$_POST`时就是这种情况。但是可以通过在一个HTML表单中创建数组的方式来做更多事情，而它将是更庞大的`$_POST`数组中的一部分（因此`$_POST`应为多维数组）。

对于这项功能，最符合逻辑的使用就是对于复选框的处理，用户可以选择多个相关的选项（参见图7-14）。复选框的HTML源代码如下：

```
<input type="checkbox"
name="topping" value="Ham" />
```

Pizza Toppings: ☐ Extra Tomato ☐ Ham ☐ Sausage ☐ Pepperoni
☐ Black Olives ☐ Turnips ☐ Kumquats

图7-14　在HTML表单中的复选框，显示多个可选项

问题是为了向PHP脚本发送多个值，每个表单元素必须有一个唯一的名称。如果创建了多个复选框，每个的名称都是`topping`，那么只有最后一个复选框的值将被PHP脚本获取。如果为每个复选框（Ham、Tomato、Black-Olives等）都创建一个唯一的值，会是一件非常烦琐的工作。

解决这个问题的方法就是使用数组，如下面的例子所示。

⇒ 在HTML表单中创建数组

(1) 在文本编辑器或者IDE中开启一个新的文档，命名为`event.html`（参看脚本7-8）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Add an Event</title>
</head>
<body>
<!-- 脚本7-8 - event.html -->
<div><p>Use this form to add an event:</p>
```

脚本7-8　该HTML表单用一个数组来处理多选框的名称

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Add an Event</title>
7   </head>
8   <body>
9   <!-- 脚本7-8 - event.html -->
10  <div><p>Use this form to add an event:</p>
11
12  <form action="event.php" method="post">
13
14    <p>Event Name: <input type="text" name="name" size="30" /></p>
15    <p>Event Days:
16    <input type="checkbox" name="weekdays[ ]" value="Sunday" /> Sun
17    <input type="checkbox" name="weekdays[ ]" value="Monday" /> Mon
18    <input type="checkbox" name="weekdays[ ]" value="Tuesday" /> Tue
19    <input type="checkbox" name="weekdays[ ]" value="Wednesday" /> Wed
20    <input type="checkbox" name="weekdays[ ]" value="Thursday" /> Thu
21    <input type="checkbox" name="weekdays[ ]" value="Friday" /> Fri
22    <input type="checkbox" name="weekdays[ ]" value="Saturday" /> Sat
23    </p>
24    <input type="submit" name="submit" value="Add the Event!" />
25
26  </form>
27  </div>
28  </body>
29  </html>
```

(2) 开始HTML表单部分：

```
<form action="event.php" method="post">
```

这个表单会提交到与这个HTML页面位于同一个文件夹下的event.php脚本。

(3) 为事件名称创建一个文本输入框：

```
<p>Event Name: <input type="text" name="name" size="30" /></p>
```

这个示例让用户能够输入事件的名称和事件发生在周几。

(4) 创建一周七天的复选框：

```
<p>Event Days:
<input type="checkbox" name="weekdays[ ]" value="Sunday" /> Sun
<input type="checkbox" name="weekdays[ ]" value="Monday" /> Mon
<input type="checkbox" name="weekdays[ ]" value="Tuesday" /> Tue
<input type="checkbox" name="weekdays[ ]" value="Wednesday" /> Wed
<input type="checkbox" name="weekdays[ ]" value="Thursday" /> Thu
<input type="checkbox" name="weekdays[ ]" value="Friday" /> Fri
<input type="checkbox" name="weekdays[ ]" value="Saturday"
/> Sat
</p>
```

所有这些复选框都使用days[]作为name的值，这也在PHP脚本中创建了一个$_POST['days']数组。每个复选框中的value属性是不同的，分别对应一周的七天。

(5) 完成HTML表单：

```
<input type="submit" name="submit" value="Add the Event!" />
</form>
```

(6) 完成HTML页面：

```
</div>
</body>
</html>
```

(7) 将页面保存为event.html，并放置在启用了PHP服务器上适当的目录中。

我们还需要编写一个event.php页面来处理这个HTML表单。

⇛ 处理HTML表单

(1) 在文本编辑器或者IDE中开启一个新的文档，命名为event.php（参看脚本7-9）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Add an Event</title>
</head>
<body>
```

脚本7-9　该PHP脚本获取到了由$_POST['days']的值构成的数组

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html;
      charset=utf-8"/>
6     <title>Add an Event</title>
7   </head>
8   <body>
9   <?php // 脚本7-9 - event.php
10  /* 脚本处理HTML表单。*/
11
12  // 可以在此进行错误处理。
13
14  // 打印文本：
15  print "<p>You want to add an event called <b>{$_POST['name']}</b> which
    takes place on: <br/>";
16
17  // 条件语句，检验是否选中选项：
18  if (isset($_POST['days']) AND is_array($_POST['days'])) {
19
20    foreach ($_POST['days'] as $day) {
21      print "$day<br/>\n";
22    }
23
```

```
24  } else {
25    print 'Please select at least one weekday for this event!';
26  }
27
28  // 完成这个段落:
29  print '</p>';
30  ?>
31  </body>
32  </html>
```

(2) 创建初始的PHP标签，添加错误管理（如果需要的话），并且打印一个介绍性的消息：

```
<?php // 脚本7-9 - event.php
print "<p>You want to add an event called <b>{$_POST['name']}</b>
→which takes place on: <br/>";
```

print行打印出事件名称的值。在该脚本的实用版本中，应该添加一个条件用来验证是否输入了值（参见第6章）。

(3) 添加一个条件语句用来检验至少有一个选项被选中：

```
if (isset($_POST['days']) AND is_array($_POST['days'])) {
```

如果所有复选框都没有被点击，那么$_POST['days']将没有值存在。为了避免由于引用一个不存在的值而引发的错误，条件的第一个部分将用来检验$_POST['days']是否被设置。

条件的第二个部分——确认$_POST['days']为一个数组，两部分条件都为TRUE时，整个条件才为TRUE。这一步非常必要，这是因为当foreach循环获得的变量不是数组的时候将会产生错误（参见图7-15）。

Warning: Invalid argument supplied for foreach() in **/Users/larryullman/Sites/phpvqs4/event.php** on line **16**

图7-15 由于foreach循环的变量不是数组而产生的常见问题

(4) 打印每个选中的选项：

```
foreach ($_POST['days'] as $day) {
  print "$day<br/>\n";
}
```

可以通过对数组$_POST['days']运行foreach循环进行遍历的方式打印出每个所选的选项。该数组包含每个被选中的复选框中的值（来自于HTML表单的输入，如Monday、Tuesday，等等）。

(5) 完成is_array()条件：

```
} else {
  print 'Please select at least one weekday for this event!';
}
```

如果没有选项被选中，那么isset() AND is_array()条件将为FALSE，并且会打印这条消息。

(6) 完成主要段落部分、PHP代码部分以及HTML页面：

```
print '</p>';
?>
</body>
</html>
```

(7) 将页面保存为`event.php`，放置在同`event.html`相同的目录中，并且在Web浏览器中进行测试（参见图7-16、图7-17和图7-18）。

Use this form to add an event:

Event Name: Training Seminar

Event Days: ☐ Sun ☐ Mon ☑ Tue ☑ Wed ☑ Thu ☐ Fri ☐ Sat

Add the Event!

图7-16　有复选框的HTML表单

You want to add an event called **Training Seminar** which takes place on:
Tuesday
Wednesday
Thursday

图7-17　HTML表单的处理结果

You want to add an event called **Training Seminar** which takes place on:
Please select at least one weekday for this event!

图7-18　如果用户没有选择任何复选框，将看到如图所示的信息

✓提示

- 这里诠释的技术同样能够让用户用来对下拉菜单进行多项选择。只需要为菜单使用诸如`something[]`的语法进行命名即可，然后PHP脚本将获得`$_POST['something']`中的每个选项。

list()函数

`list()`函数用来将数组元素的值赋给单独的变量。让我们从下面的示例开始：

```
$date = array('Thursday', 23, 'October');
list($weekday, $day, $month) = $date;
```

现在`$weekday`变量有了`Thursday`这个值，`$day`变量的值则为`23`，`$month`变量的值为`October`。

在使用`list()`时要注意两点。首先，`list()`只对数值型索引并从0开始索引的数组有用。其次，当使用`list()`函数时，必须确认接受每一个数组元素。但是可以用空值的方式忽略元素：

```
list ($weekday, , $month) = $date;
```

或者

```
list (, , $month) = $date;
```

但是不能这样做：

```
list($weekday, $month) = $date;
```

`list()`函数通常用来从数据库中检索值。

7.9　回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

7.9.1　回顾

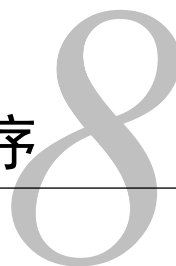
- 索引数组与关联数组之间的区别是什么？
- 什么时候应该用引号将数组的索引和值括起来？什么时候不应该用？
- 如何打印数组中的某个元素？如何打印数组中的所有元素？
- 如果在添加数组元素时不使用方括号，会发生什么？
- 哪个函数返回数组中元素的个数？
- 打印数组元素时，什么时候必须用花括号括起来？
- `sort()`和`asort()`函数的区别是什么？`sort()`和`rsort()`函数的区别是什么？
- `explode()`函数的语法是什么？`implode()`函数的语法是什么？如果你忘记了，请查看PHP手册。

7.9.2　实践

- 查找PHP手册中与数组有关的函数。研究一下其他常用的数组函数。我建议你熟悉一下`array_key_exists()`、`array_search()`和`in_array()`。
- 重写`soups2.php`，显示数组元素的个数，这次不要使用单独的变量。提示：在`print`语句中串联`count()`函数。
- 创建一个脚本，建立并显示一个多维数组（或任意一种数组）。
- 重写`list.php`，用`foreach`代替`implode()`，但依旧在浏览器中按顺序每行打印一个单词。加入表单验证功能，以便只在存在实际字符串值的时候对其进行解析和排序。

第 8 章

创建Web应用程序



本章内容

- ❑ 创建模板
- ❑ 使用外部文件
- ❑ 使用常量
- ❑ 使用日期和时间
- ❑ 再谈使用PHP处理HTML表单
- ❑ 使表单更具粘性
- ❑ 发送Email
- ❑ 输出缓冲
- ❑ 处理HTTP头
- ❑ 回顾和实践

本书已经介绍了很多PHP编程的基础知识，现在就可以将这些技术组合到一起，尝试创建一个实际的Web应用程序。本章将讨论更多函数和技巧，让Web站点更加专业、功能更丰富并且更易于维护。

首先，将讨论如何使用外部文件将Web页面划分为独立的部分（在一定程度上能够将逻辑和格式分开）。然后，介绍PHP中的常量和特殊数据类型。此处，我们还将介绍PHP的一些与日期和时间相关的函数。

本章中有两个主题用来讨论非常实用的技术：使用同一个页面同时显示和处理HTML表单，以及让表单记住用户提交的值。此后，还将看到PHP能够非常容易地发送电子邮件。最后，本章还将讨论更为高级的主题，即输出缓冲和使用HTTP头。

8.1 创建模板

迄今为止，每一个示例都有一页脚本专门处理HTML表单、对数组进行排序或执行运算。然而，在开始开发多页面的Web站点（称作Web应用程序）时，在大量的页面中重复常见的元素就会变得不切实际。

在更复杂的Web站点中，有些特性（如HTML设计）会在站点中的每个页面里都用到。可以将这些元素放到每一个独立的页面中，但当需要进行修改时，又必须一次又一次地完成修改。创建模板可以将重复的内容和特定于页面的素材分开，这样就节省了不少时间。例如，某个Web站点可能会拥有导航、版权和其他特性，这些特性在多个页面中重复使用（参见图8-1和图8-2）。

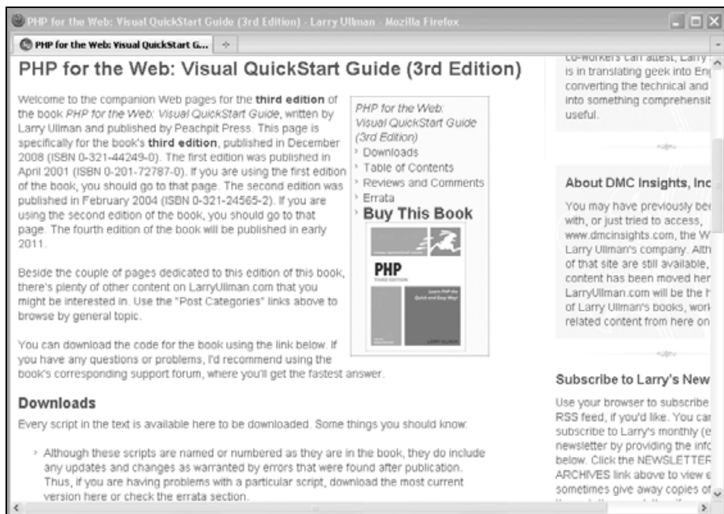


图8-1　本书第3版主页的左栏显示的是特定于页面的内容，右栏是一些公共内容

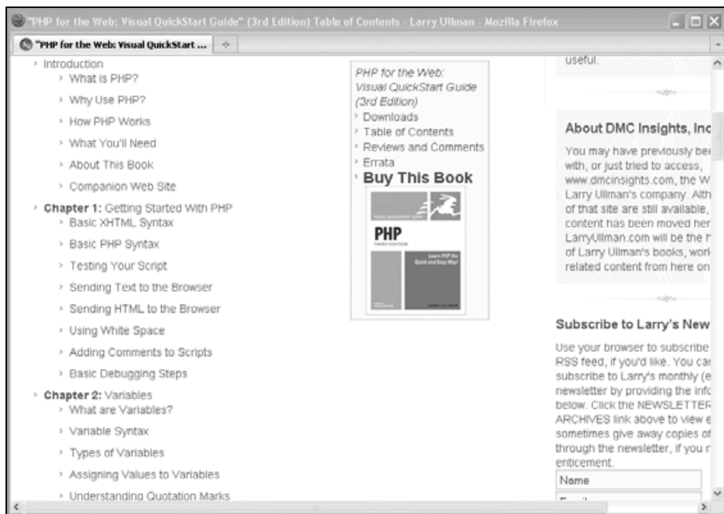


图8-2　内容页的表格使用了一些与主页（参见图8-1）相同的元素，这得益于模板的使用

当你第一次开发动态Web站点时，创建和使用模板会非常困难。使用模板的关键在于要创建一个原型，并将其切分成不同的部分。然后使用8.2节中介绍的PHP函数，这些重复的部分就可以

很方便地包含到每个页面中，然后逐页面地生成新内容。下面创建本章示例中要使用的模板，首先建立一个原型。这个示例的布局（参见图8-3）是由Six Shooter Media（www.sixshootermedia.com）的James Koster设计的，并且已获得了使用许可。



图8-3　本章示例的页面设计，一个单独的、静态的HTML页面。

⇛ 创建布局模型

(1) 在文本编辑器或IDE中创建一个新的HTML文档，命名为template.html(参看脚本8-1)：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.
→w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <title>Raise High the Roof Beam! A J.D. Salinger Fan Club</title>
  <meta http-equiv="content-type" content="text/html;
  →charset=utf-8" />
```

脚本8-1　该脚本展现了网站每个页面的基本外观

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/
    DTD/xhtml11.dtd">
2   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
3   <head>
4     <title>Raise High the Roof Beam! A J.D. Salinger Fan Club</title>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <link rel="stylesheet" href="css/1.css" type="text/css"
      media="screen,projection" />
7   </head>
8   <body>
9   <div id="wrapper">
```

```
10
11    <div id="header">
12      <p class="description">A J.D. Salinger Fan Club</p>
13      <h1><a href="index.php">Raise High the Roof Beam!</a></h1>
14      <ul id="nav">
15        <li><a href="books.php"> Books</a></li>
16        <li><a href="#">Stories</a></li>
17        <li><a href="#">Quotes</a></li>
18        <li><a href="login.php"> Login</a></li>
19        <li><a href="register.php"> Register</a></li>
20      </ul>
21    </div><!-- 页头 -->
22
23    <div id="sidebar">
24      <h2>Favorite Quotes</h2>
25        <p class="news">I don't exactly know what I mean by that, but
          I mean it.<br/>- <em>The Catcher in the Rye</em></p>
26        <p class="news">I privately say to you, old friend... please
          accept from me this unpretentious bouquet of early-blooming
          parentheses: (((()))).<br/>- <em>Raise High the Roof Beam,
          Carpenters and Seymour: An Introduction</em></p>
27    </div><!-- 边栏 -->
28
29    <div id="content">
30      <!-- 可变内容开始。-->
31      <h2>Welcome to a J.D. Salinger Fan Club</h2>
32      <p>Lorem ipsum dolor sit amet,
        consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et
        dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
        laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
        reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
        Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
        deserunt mollit anim id est laborum.</p>
33      <h2>Another Header</h2>
34      <p>Lorem ipsum dolor sit amet,
        consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et
        dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
        laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
        reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
        Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
        deserunt mollit anim id est laborum.</p>
35      <!-- 可变内容结束。-->
36    </div><!-- 内容 -->
37
38    <div id="footer">
39      <p>Template design by <a href="http://www.sixshootermedia.com">
        Six Shooter Media</a>.</p>
40      <p>&copy; 2011</p>
41    </div><!-- 页脚 -->
42
43  </div><!-- 完成页面 -->
44  </body>
45  </html>
```

开发任何模板系统的第一步都是创建一个模型文档——一个展示了基本的页面样子的实例。一旦创建好模型文档，就可以将其划分成块了。

(2) 添加CSS代码：

```
<link rel="stylesheet" href="css/1.css" type="text/css"
→media="screen,projection" />
```

对于这个应用程序，将使用CSS进行所有的格式化和布局管理。CSS存储在一个外部文件中，通过`link`标签嵌入到页面中。该文件被简单命名为`1.css`，存储在`css`文件夹下。

注意，你需要从本书的网站（www.LarryUllman.com）上下载这个CSS文件。这个文件是本书可下载代码的一部分。

(3) 关闭HTML `head`，打开`body`，创建一个`div`标签`id`为`wrapper`：

```
</head>
<body>
<div id="wrapper">
```

如今，很多设计将页面的全部内容放在主`div`标签之中，这样可以很方便地在浏览器窗口中格式化所有内容。

(4) 创建页面的`header`：

```
<div id="header">
  <p class="description">A J.D.Salinger Fan Club</p>
  <h1><a href="index.php">Raise High the Roof Beam!</a></h1>
  <ul id="nav">
  <li><a href="books.php"> Books</a></li>
  <li><a href="#">Stories</a></li>
  <li><a href="#">Quotes</a></li>
  <li><a href="login.php"> Login</a></li>
  <li><a href="register.php"> Register</a></li>
  </ul>
</div><!-- 页头 -->
```

`header`区域（CSS代码中也已定义）创建标语和主导航链接，这些链接可以导航到这个Web应用程序的其他页面。这些链接引用了4个PHP脚本，本章会教你编写这些脚本。

(5) 创建页面的边栏：

```
<div id="sidebar">
  <h2>Favorite Quotes</h2>
  <p class="news">I don't exactly know what I mean by that, but I mean it.
  →<br/>- <em>The Catcher in the Rye</em></p>
  <p class="news">I privately say to you, old friend...
  →please accept from me this unpretentious bouquet of early-blooming parentheses:
  →(((()))).<br/>- <em>Raise High the Roof Beam, Carpenters and Seymour:
  →An Introduction</em></p>
</div><!-- 边栏 -->
```

在这个最初的模板中，边栏可以用于发布最新消息、链接次要网页、放置搜索框，等等。就示例网站而言，边栏用于突出一些引文。

(6) 首先，标记特定于页面内容的开始：

```
<div id="content">
<!-- 可变内容开始。-->
```

到这条注释为止的所有内容在Web应用程序中的每个页面都保持不变。为了明确特定页面内容的起始点（主要是为自己方便），这里添加了一个HTML注释。实际上，你可以在这个模板中看到很多注释，指出了这些HTML对应着页面中的哪一部分。但也不要在HTML或PHP脚本中添加过多的注释信息。

在这之前是content区域的开始。该区域在CSS代码中进行了定义，并恰当地定义了页面主内容部分的格式。换句话说，每个页面中的内容都将放置在一个id为content的div中。

(7) 创建页面的内容：

```
<h2>Welcome to a J.D. Salinger Fan Club</h2>
<p>Lorem ipsum dolor sit amet...</p>
```

对于这个原型来说，内容只是一系列标题和文本（实际脚本中的内容要比我在此添加的东西多得多）。

(8) 标记可改变内容的结束：

```
<!-- 可变内容结束。-->
</div><!-- 内容 -->
```

第7步代码显示的文本是唯一会根据页面的不同而产生改变的文本。前面的一个HTML注释指出了这个区域的起始位置，这个注释指出了其结束位置。

(9) 添加footer：

```
<div id="footer">
  <p>Template design by <a href="http://www.sixshootermedia.
  →com">Six Shooter Media</a>.</p>
  <p>&copy; 2011</p>
</div><!-- 页脚 -->
```

footer包括一些版权声明信息。

(10) 完成HTML页面:

```
</div><!-- 完成页面 -->
</body>
</html>
```

注意：HTML注释标明了关闭的是哪个div标签，这有助于修改和维护模板。

(11) 将文件保存为template.html，并在Web浏览器中测试（参见图8-3）。

以自己喜欢的方式完成了一个原型之后，就可以将其划分成多个部分，用以生成模板系统。

⇛ 创建页头文件

(1) 如果尚未打开template.html（参看脚本8-1），在文本编辑器或IDE中打开它。

(2) 选中从HTML代码开头到HTML注释<!-- 可变内容开始 -->之间的所有内容（参见图8-4）。

8



图8-4　使用示例文件，选中并复制前面若干行代码，用于创建页头文件

用HTML注释指出特定于页面的内容，这样做的优点之一在于可以简化切分模型的工作。

(3) 复制代码

使用Edit菜单或快捷键（Windows上是Ctrl+C、Macintosh上是Command+C），将选中的代码复制到计算机的临时内存（也就是剪贴板）中。

(4) 在文本编辑器或IDE中创建一个新的空白文档，命名为`header.html`。

(5) 将复制的文本粘贴到文档中（参看脚本8-2）。

脚本8-2　这是一个基本的页头文件，创建了HTML `head`信息（包含CSS）和`body`的起始标签

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
2   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
3   <head>
4     <title>Raise High the Roof Beam!
      A J.D. Salinger Fan Club</title>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <link rel="stylesheet" href="css/1.css" type="text/css"
      media="screen,projection" />
7   </head>
8   <body>
9   <div id="wrapper">
10
11    <div id="header">
12      <p class="description">A J.D. Salinger Fan Club</p>
13      <h1><a href="index.php">Raise High the Roof Beam!</a></h1>
14      <ul id="nav">
15        <li><a href="books.php">Books
          </a></li>
16        <li><a href="#">Stories</a></li>
17        <li><a href="#">Quotes</a></li>
18        <li><a href="login.php">Login
          </a></li>
19        <li><a href="register.php">
          Register</a></li>
20      </ul>
21    </div><!-- 页头 -->
22
23    <div id="sidebar">
24      <h2>Favorite Quotes</h2>
25        <p class="news">I don't exactly know what I mean by that, but I mean it.
          <br/>- <em>The Catcher in the Rye</em></p>
26        <p class="news">I privately say to you, old friend... please accept from me
          this unpretentious bouquet of early-blooming parentheses: (((()))).<br/>-
          <em>Raise High the Roof Beam, Carpenters and Seymour: An Introduction
          </em></p>
27    </div><!-- 边栏 -->
28
29    <div id="content">
30      <!-- 可变内容开始。-->
31      <!-- 脚本8-2 - header.html -->
```

使用Edit菜单或快捷键（Windows上是Ctrl+V，Macintosh上是Command+V），将所有选中的代码复制到新文档中。

(6) 将文件保存为`header.html`。

既然页头文件已经创建好，就可以用同样的步骤创建页脚文件。

⇛ 创建页脚文件

(1) 如果尚未打开`layout.html`（参看脚本8-1），在文本编辑器或IDE中打开它。

(2) 选择从HTML注释`<!-- 可变内容结束 -->`到脚本结束之间的所有内容（参见图8-5）。

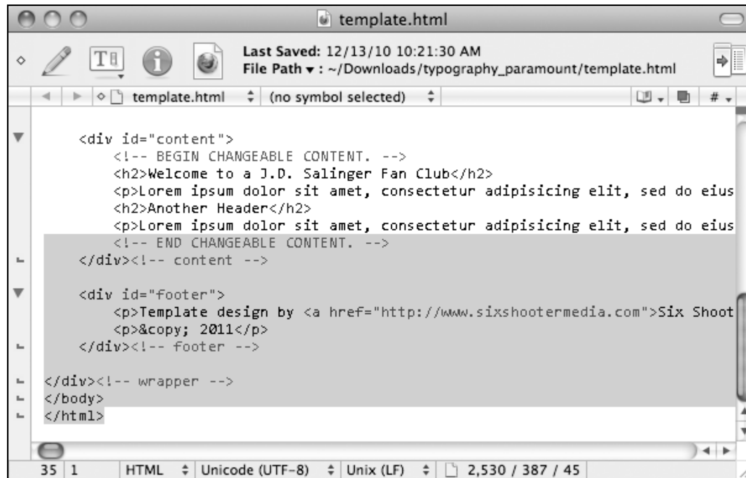


图8-5　继续使用该示例文件，选中并复制最后的若干行代码，用于创建页脚

(3) 复制这部分代码。

(4) 在文本编辑器或IDE中创建一个新的空白文档，命名为`footer.html`。

(5) 将复制的文本粘贴到文档中（参看脚本8-3）。

脚本8-3　这是一个基本的页脚文件，创建了导航栏和HTML页面的结束部分

```
1        <!-- 脚本8-3 - footer.html -->
2        <!-- 可变内容结束。-->
3      </div><!-- 内容 -->
4
5      <div id="footer">
6        <p>Template design by <a href="http://www.sixshootermedia.com">
         Six Shooter Media</a>.</p>
7        <p>&copy; 2011</p>
8      </div><!-- 页脚 -->
9
10   </div><!-- 完成页面 -->
11   </body>
12   </html>
```

(6) 将文件保存为`footer.html`。

✓提示

- 在PHP中还可以使用更为复杂的模板系统，将设计与逻辑分开。Smarty（www.smarty.net）可能是最有名的模板系统。
- 尽管该示例使用CSS进行布局，但也可以使用表格来代替（参见图8-6）。页头文件可能需要同时打开HTML页和表格。每个内容页都会创建其特定内容，然后页脚文件需要同时完成表格和HTML页面。要将其转换为模板，需要将Page-specific content goes here之前的代码复制到页头文件中，而将其后的所有代码复制到页脚文件中。

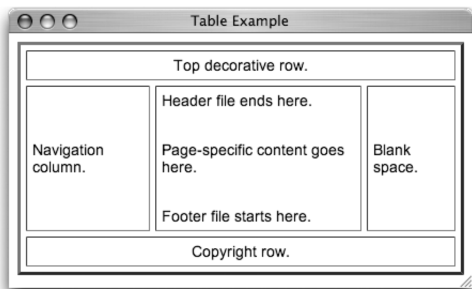


图8-6　这个简单的示例展示了如何在模板文件中使用表格进行设计

CSS　模　板

一段时间以来，层叠样式表（CSS）已经成为万维网中越来越重要的一部分了。它们最初只是用于装饰（字体大小、颜色等），但现在CSS常用于替代表格进行页面布局管理。本章中的Web应用程序都是采用的CSS方式。

该示例在页面中定义了4个区域——`header`、`sidebar`、`content`和`footer`。`content`区域随着页面而变。其他区域包含了标准的项（如导航链接），它会出现在应用程序的每个页面中。

要明确的是：PHP与CSS的关系和PHP与HTML的关系一样——PHP运行在服务端，而HTML和CSS是在浏览器上运行的。可以用PHP来生成CSS，就像使用PHP来生成HTML一样。不过在这个示例中，CSS是硬编码到HTML文档的`head`元素中的。

8.2　使用外部文件

正如8.1节所述，可以将页面划分成特定的元素，然后使用特定的函数将它们合并到主PHP页面中，这样可以节省开发时间。需要使用的函数有`include()`和`require()`：

```
include ('file.php');
require ('file.html');
```

这两个函数的工作方式相同，它们的区别也不是很大：如果`include()`函数失败了，PHP脚本会生成一个警告（参见图8-7），但继续运行。相反，如果`require()`失败了，它会终止脚本的执行（参见图8-8）。

这两个函数都用来做些什么呢？`include()`和`require()`都会将所引用的文件合并到主文件中（为了清晰起见，本章将包含有`include()`或`require()`代码行的文件称作包含文件或父文件）。这两个函数运行的结果将是一样的，就像被包含的代码是父文件的一部分一样。

包含文件的效果，就好像被包含文件的内容本来就在父脚本中一样，理解这一点是用好这个函数的关键。这就意味着被包含文件中没有放在PHP标签里的任何代码都将被当成HTML。而且，无论被包含的文件扩展名是什么结果都是一样的（因为这个文件只是包含它的文件的一个扩展）。



图8-7　当`include()`失败时，会产生警告，但脚本能够继续执行

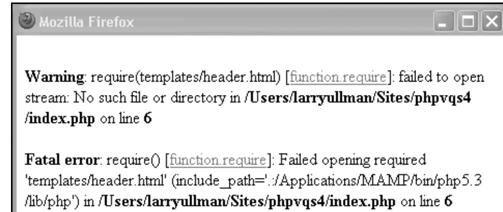


图8-8　当`require()`函数调用失败时，同时产生警告和错误，并且脚本停止执行

使用被包含文件有很多原因。可以将自己定义的函数放在一个通用文件中（有关编写自有函数的更多信息，请参见第10章）。还可以将数据库访问信息放置在一个配置文件中（参见第12章）。不过，首先我们还是包含8.1节创建的模板文件，以便每个页面都能具备一致的设计风格。

⇒ 使用外部文件

(1) 在文本编辑器或IDE中创建一个新的文档，命名为`index.php`。

(2) 在文档起始处输入PHP初始标签，并添加一些注释（参看脚本8-4）：

```
<?php // 脚本8-4 - index.php
/* 这是网站的主页。
使用模板创建页面布局。*/
```

脚本8-4　创建好两个被包含文件之后，可以使用`include()`函数将其合并到父文件中，以在运行时创建整个页面

```
1   <?php // 脚本8-4 - index.php
2   /* 这是网站的主页。
3   使用模板创建页面布局。*/
4   
5   // 包含页头文件：
6   include('templates/header.html');
7   // 关闭PHP节创建HTML内容：
8   ?>
9   
10  <h2>Welcome to a J.D. Salinger Fan Club</h2>
11  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
    incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute
    irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
    pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
    deserunt mollit anim id est laborum.</p>
12  <h2>Another Header</h2>
13  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
    incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
```

```
   exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute
   irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
   pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
   deserunt mollit anim id est laborum.</p>
14
15 <?php // 创建另一个PHP节。
16 include('templates/footer.html'); // Include the footer.
17 ?>
```

注意，有了模板系统后，脚本的第一行就是PHP标签。这里无需再以HTML代码开始，因为它们都被放到了header.html文件中。

(3) 如果需要的话，添加处理错误消息的管理。

这个主题已经在第3章中讨论过，在脚本中，可以处理错误也可以不处理错误。更多信息请参见第3章，这将是在本章中最后一次着重提到这一点。

(4) 包含页头文件：

```
include('templates/header.html');
```

要使用模板系统，需要在这里通过调用include()函数来包含页头文件。由于页头文件中只包含HTML，因此其中所有的内容会直接发送到Web浏览器，就好像这些内容就是当前文件的一部分。这一行使用相对路径来引用包含文件（参见框注"文件导航和站点结构"），并假设页头文件存储在templates目录中。

(5) 关闭PHP节并创建特定页面的内容：

```
?>
<h2>Welcome to a J.D. Salinger Fan Club</h2>
<p>Lorem ipsum dolor sit amet...</p>
```

由于该页面主要是标准的HTML，因此这里暂时关闭PHP节并输入HTML（而不是使用print将HTML发送到Web浏览器）。同时，这里包含的文字相对于脚本本身来说没什么特殊意义。

(6) 创建另一个PHP节并请求页脚文件：

```
<?php
include('templates/footer.html');
?>
```

为了完成该页面，还需要包含页脚文件（用于显示导航栏并关闭HTML代码）。要完成这一任务，需要创建一个新的PHP节——在一个脚本中可以有多个PHP节——并再次调用include()函数。

(7) 将文件保存为index.php。

(8) 在启用了PHP的计算机或服务器上的Web文档目录中，创建一个名为templates的子目录。

进一步分离设计元素和主内容，将页头和页脚文件放到它们自己的目录中。

(9) 将header.html和footer.html放在刚才创建的templates目录中。

(10) 将index.php放置在和templates文件夹同级的目录中。

(11) 在启用了PHP的计算机或Web服务器的主Web文档目录中，创建一个名为css的子目录。

这个文件夹用于保存CSS脚本。

(12) 从本书的随时代码中找到1.css脚本，保存在css文件夹下（参见图8-9）。

虽然在头文件中包括CSS脚本，但对这个脚本的引用必须相对于index.php。毕竟，这个页面会包含header.html。

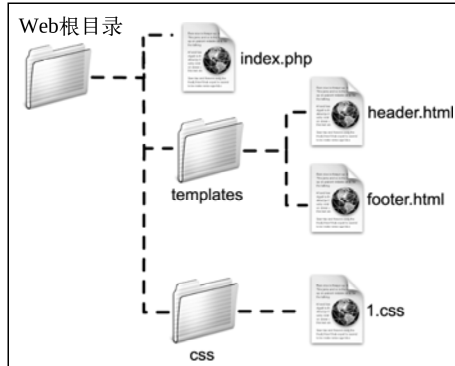


图8-9　在启用了PHP的服务器上，4个文件和2个文件夹的组织结构图

索引页和两个HTML页面在计算机上的相对位置必须正确，这样代码才能正常工作。

(13) 在Web浏览器中运行index.php（参见图8-10）。

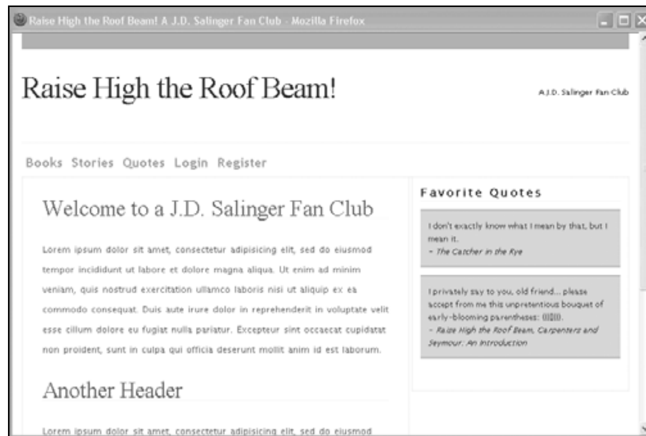


图8-10　该页面是使用被包含文件动态生成的

结果页看上去应该和原始布局完全一致（参见图8-3）。

(14) 在Web浏览器中查看页面源代码。

除了为脚本名字添加的注释和数字，源代码应该与template.html脚本（参看脚本8-1）中的源代码一致。

✓提示

❑ 模板系统中所有的3个文件（header.html、footer.html和index.php）必须使用相

同的编码，以避免出现问题（关于编码的更多信息，请参见第1章）。每个文件的编码还必须和HTML代码中建立的编码相匹配。

- 在使用`require()`和`include()`函数时，可以使用圆括号，也可以不使用圆括号：

```
require 'filename.html';
```

- 有时也可以使用变量来存放要包含的文件名字：

```
require $filename;
```

- `include()`和`require()`都具有变种：`include_once()`和`require_once()`。这两个函数与前两者一一对应，但它们只允许同一个文件被包含一次（在同一个父文件中）通常应该尽量避免使用它们，因为这两个函数会影响脚本的性能。
- 如果PHP段只包含一行语句，一般将这条语句与PHP标签写在同一行：

```
<?php include 'filename.html'; ?>
```

- 如果看到了类似于图8-7和图8-8中的错误消息，则表示父脚本无法定位被包含文件。这个问题很可能是由于被包含文件名拼写错误或路径错误造成的（例如，使用`header.html`替换了`templates/header.html`）。
- 如果呈现的网站没有显示CSS样式，HTML页面无法找到相关文件。请确认文件是否放在正确的文件夹下，文件名是否正确，引用是否使用的是相对于`index.php`的链接。
- 文件的扩展名对于被包含文件来说并不是很重要，因为它们不会直接执行。一般的经验法则是，可以安全地为只包含或主要包含HTML的被包含文件使用`.html`扩展名（该扩展名指出这是一个HTML相关的文件），以及为只包含或主要包含PHP的文件使用`.php`扩展名。一些开发者使用`.inc`扩展名（表示include），但这样做会带来相关的安全风险。因此，请为任何包含有敏感信息（如数据库访问参数）的文件使用`.php`扩展名。当然，对于需要直接执行的PHP脚本，也要使用`.php`扩展名。
- 外部文件的另一种比较好的使用方式是将错误设置代码放在其中，这样设置的变更就能应用于Web站点中的每一个页面了。

文件导航和站点结构

为了使用外部文件，需要理解计算机或服务器上的文件导航。和在HTML链接或图片中引用Web站点中的其他页面一样，必须为父文件指出被包含脚本。这可以使用绝对路径或相对路径。绝对路径是一个特定的地址，如下所示：

```
include('C:\inetpub\wwwfiles\file.php');
include('/Users/larry/Sites/file.php');
```

只要被包含文件没有移动过，绝对路径就永远有效。

相对路径指出了被包含文件相对于父文件的位置关系。下面的示例假设被包含文件和父文件位于同一个目录中：

```
include('file.php');
```

```
include('./file.php');
```

被包含文件也可以位于比父文件低一级的目录中，就像本章示例中使用的那样（参见图8-9）：

```
include('templates/header.html')
```

或者，被包含文件可以位于父文件的上一级目录中：

```
include('../file.php');
```

最后是对于站点结构的说明：一旦将Web应用程序划分成多个小块，就需要开始考虑如何将这些文件放置到适当的目录中。复杂的站点可能会拥有一个主文件夹、一个用于放置图片的文件夹、一个用于放置管理文件的文件夹，以及一个专门放置模板和被包含文件的特殊目录。只要在`include()`或`require()`语句中正确地引用文件，使用结构化的应用程序就可以工作得很好，并且维护起来也更方便。

8.3　使用常量

本书已经讨论了PHP中的很多数据类型：基本数值、字符串和数组。常量是另外一种数据类型，但和变量不同，它们在脚本的执行过程中会一直保持其初始值。一旦设定了常量值，就不能再更改它了。

只能通过为其赋值来创建常量。和变量使用赋值运算符（`=`）进行赋值不同，常量的赋值需要使用`define()`函数：

```
define('CONSTANT_NAME', value);
```

请注意，一条经验法则是，全部使用大写字母来命名常量，尽管并不是必须这样做。更重要的是，常量不像变量那样在字首使用美元符号（因为常量不是变量）。这里有两个常量：

```
define ('PI', 3.14);
define ('CURRENCY', 'euros');
```

对于*`value`*来说，字符串需要用引号括起来，而数值不需要。

常量的引用通常很简单：

```
print CURRENCY;
number_format(PI, 1);
```

但在引号内部使用常量时有些复杂，这里无法打印出单引号或双引号内部的常量值，如（参见图8-11）：

```
print "The cost is 468 CURRENCY";
print 'The cost is 468 CURRENCY';
```

The cost is 468 CURRENCY

图8-11　无法打印出单引号或双引号内部的常量值

这里需要使用连接或使用多个print语句：

```
print 'The cost is 468 ' . CURRENCY;
```

或

```
print 'The cost is 468 ';
print CURRENCY;
```

请不要混淆，和用于创建常量的define()函数一同使用的还有defined()函数，如果提交的常量已定义，则这个函数会返回TRUE。defined()函数常常用在条件语句中：

```
if (defined('CONSTANT_NAME')) { ...
```

作为使用常量的一个示例，我们将为前面的示例程序添加一种能够为每个页面显示不同标题（将出现在浏览器窗口的顶部）的功能。为了完成这一功能，需要在父脚本中定义一个常量，然后在页头文件中将父脚本打印出来。这种技术是可行的，只要在include()语句或require()语句中恰当地引用了这些文件，结构化的应用程序就能运行得很好，并带来易于维护等优点。

⇒ 使用常量的步骤

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为books.php（参看脚本8-5）：

```
<?php // 脚本8-5 - books.php
```

脚本8-5 这个脚本不但使用和index.php（参看脚本8-4）相同的模板系统，而且还使用了一个用于标识页面标题的常量

```
1   <?php // 脚本8-5 - books.php
2   /* 这个页面显示J.D. Salinger的著作。*/
3
4   // 设置页面标题并包含页头文件：
5   define ('TITLE', 'Books by J.D. Salinger');
6   include('templates/header.html');
7
8   // 关闭PHP节创建HTML内容：
9   ?>
10
11  <h2>J.D. Salinger's Books</h2>
12  <ul>
13    <li>The Catcher in the Rye</li>
14    <li>Nine Stories</li>
15    <li>Franny and Zooey</li>
16    <li>Raise High the Roof Beam, Carpenters and Seymour: An Introduction</li>
17  </ul>
18
19  <?php include('templates/footer.html'); ?>
```

(2) 将页面标题定义为常量：

```
define ('TITLE', 'Books by J.D. Salinger');
```

这里定义了一个常量，名为TITLE，并为其赋值Books by J.D. Salinger。

(3) 包含页头文件：

```
include('templates/header.html');
```

该脚本使用和所有其他脚本相同的页头文件，但稍后将对其进行少许修改，将常量引入进来。

(4) 关闭PHP节，创建HTML：

```
?>
<h2>J.D. Salinger's Books</h2>
<ul>
  <li>The Catcher in the Rye</li>
  <li>Nine Stories</li>
  <li>Franny and Zooey</li>
  <li>Raise High the Roof Beam,
  →Carpenters and Seymour: An
  →Introduction</li>
</ul>
```

这里给出的内容非常简单，但正好达到适用于页面的目的。

(5) 创建一个新的PHP节并包含页脚文件：

```
<?php include('templates/footer.html'); ?>
```

在8.2节的提示中说过，由于PHP代码只有一条语句，可以将这条语句与PHP开始和关闭标签写在同一行。但要注意在要执行的代码（`include()`）和标签之间要加一个空格。

(6) 将文件保存为`books.php`。

为了使用常量，接下来还需要修改`header.html`文件。

⇛ 打印一个常量

(1) 在文本编辑器或IDE中打开`header.html`（参看脚本8-2）。

(2) 删除出现在`title`标签中的*Raise High the Roof Beam! A J. D. Salinger Fan Club*（参见第4行）。

既然页面标题将根据页面的不同有所变化，就不需要将其硬编码到页面中了。

(3) 在删除文本的地方（`title`标签中），添加下面的脚本（参看脚本8-6）：

```
<?php
if (defined('TITLE')) {
  print TITLE;
} else {
  print 'Raise High the Roof Beam! A J.D. Salinger Fan Club';
}
?>
```

脚本8-6　`header.html`文件经过了修改，这样它就可以根据常量是否存在以及常量的值来设置页面标题

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
2   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
3   <head>
```

```
4    <title><?php // 打印页面标题。
5    if (defined('TITLE')) { // 检查标题是否已定义。
6      print TITLE;
7    } else { // 标题没有定义。
8      print 'Raise High the Roof Beam! A J.D. Salinger Fan Club';
9    }
10   ?></title>
11   <meta http-equiv="content-type" content="text/html; charset=utf-8" />
12   <link rel="stylesheet" href="css/1.css" type="text/css"
     media="screen,projection" />
13 </head>
14 <body>
15 <div id="wrapper">
16
17   <div id="header">
18     <p class="description">A J.D. Salinger Fan Club</p>
19     <h1><a href="index.php">Raise High the Roof Beam!</a></h1>
20     <ul id="nav">
21       <li><a href="books.php">Books</a></li>
22       <li><a href="#">Stories</a></li>
23       <li><a href="#">Quotes</a></li>
24       <li><a href="login.php">Login</a></li>
25       <li><a href="register.php">Register</a></li>
26     </ul>
27   </div><!-- 页头 -->
28
29   <div id="sidebar">
30     <h2>Favorite Quotes</h2>
31       <p class="news">I don't exactly know what I mean by that, but I mean it.
         <br/>- <em>The Catcher in the Rye</em></p>
32       <p class="news">I privately say to you, old friend... please
         accept from me this unpretentious bouquet of early-blooming
         parentheses: (((()))).<br/>- <em>Raise High the Roof Beam,
       Carpenters and Seymour: An Introduction</em></p>
33   </div><!-- 边栏 -->
34
35   <div id="content">
36     <!-- 可变内容开始。-->
37     <!-- 脚本8-6 - header.html -->
```

为了能让PHP创建页面标题，首先需要在title标签中打开一个PHP代码节。然后使用一个条件语句检查是否已经定义了TITLE常量。如果已经定义，则打印它的值并作为页面标题。如果没有定义TITLE，则打印一个默认标题。

(4) 将文件保存为header.html。

(5) 将books.php和header.html上传到启用了PHP的服务器上。books.php这个新脚本应该和index.php位于同一目录，而header.html应该替换掉之前的版本，并和footer.html一样都位于目录templates中。

(6) 在Web浏览器中运行books.php（参见图8-12）。

(7) 在Web浏览器中查看index.php（主页）（参见图8-13）。

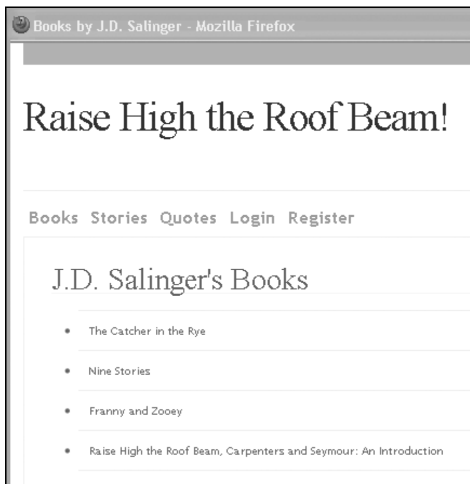


图8-12　books页面使用PHP常量来创建标题

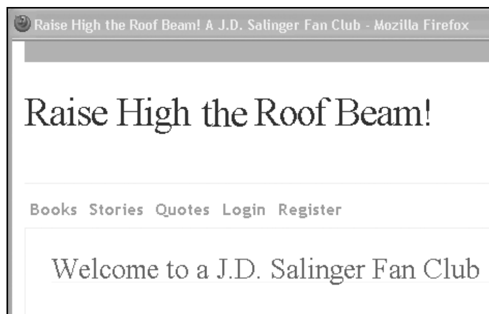


图8-13　由于index页中没有定义TITLE常量，所以该页面的标题使用了默认值（这是由于在脚本8-6中使用了条件语句）

(8) 如果有兴趣，可以在index.php中添加一个常量定义，修改其标题。

✓提示

- 除了要省略掉美元符号之外，常量的正式命名规则和变量是完全一样的。常量名称必须以字母开头，可以包含字母、数字和下划线的任意组合，并且是区分大小写的。
- PHP有很多预定义的常量。包括PHP_VERSION（正在运行的PHP的版本）和PHP_OS（服务器的操作系统）。
- 在第9章中，会介绍另外一个常量——SID（表示session的ID）。
- 使用常量的另外一个好处是，它们是全局作用域的。在阅读完第10章中讲述这一部分的内容之后，读者会对此有更多认识。
- 不仅常量的值不能修改，常量本身也不能删除。另外，一个常量只能包含一个单独的值，这和数组不同，但是和字符串或数字是一样的。

8.4　使用日期和时间

PHP具备一些用于操作日期和时间的函数，其中最重要的就是date()。date()函数做的唯一的事情就是基于提供给它的参数，返回格式化的日期和时间信息，但当了解到它有多么有用后，你一定会感到惊讶！date()函数的基本使用方法是：

```
date('formatting');
```

可以用于进行格式化的选项有很多，表8-1列出了所有的选项。这些参数还可以组合——例如，date('l F j, Y')可以返回Wednesday January 26, 2011。

表8-1　**Date()**函数格式

字　符	含　义	示　例
`Y`	4位数字表示的年份	2011
`y`	2位数字表示的年份	11
`L`	是否为闰年	1（表示“是”）
`n`	1或2位数字表示的月份	2
`m`	2位数字表示的月份	02
`F`	月份	February
`M`	3个字母表示的月份	Feb
`j`	1或2位数字表示的月份中的一天	8
`d`	2位数字表示的月份中的一天	08
`l`（小写的L）	一周中的某一天	Monday
`D`	3个字母表示的一周中的某一天	Mon
`w`	1位数字表示的一周中的某一天	0（星期日）
`z`	一年中的某一天：0~365	189
`t`	月份中有多少天	31
`S`	2个字符表示的天数英文序数词后缀	rd
`g`	小时数，1或2位数字表示的12小时制格式	6
`G`	小时数，1或2位数字表示的24小时制格式	18
`h`	小时数，2位数字表示的12小时制格式	06
`H`	小时数，2位数字表示的24小时制格式	18
`i`	分钟数	45
`s`	秒数	18
`u`	毫秒数	1234
`a`	am或pm	am
`A`	AM或PM	PM
`U`	从epoch开始的秒数	1048623008
`e`	时区	UTC
`I`（大写的i）	是否为夏令时	1（表示“是”）
`O`	与GMT之间的时差	+0600

`date()`函数可以接受另一个参数，这个参数称作时间戳（timestamp）。时间戳是一个数字，表示从1970年1月1日午夜起计算的秒数——这一时刻也称作epoch。`time()`函数可以返回当前时刻的时间戳。`mktime()`函数可以返回一个给定的时间和日期的时间戳：

```
mktime(hour, minute, second, month, day, year);
```

因此下面的代码：

```
$ts = mktime(12, 30, 0, 11, 5, 2011);
```

可以将从epoch到2011年11月5日12:30经过的秒数赋值给`$ts`变量。这个数值可以传递给`date()`函数，如下所示：

```
date('D', $ts);
```

这会返回Sat，它是用3个字母表示的一周中的某一天。

对于PHP5.1，需要在调用任何日期或时间相关的函数之前，设置服务器的时区。要完成这一任务，请使用：

```
date_default_timezone_set(timezone);
```

*`timezone`*的值是类似于`America/New_York`或`Pacific/Auckland`这样的字符串。这些值太多了，无法在这里列出（仅非洲就有超过50个时区），可以在PHP手册中查到所有的值。如果没有完成这一步，则会看到错误（参见图8-14）。

Warning: date() [function.date]: It is not safe to rely on the system's timezone settings. You are *required* to use the date.timezone setting or the date_default_timezone_set() function. In case you used any of those methods and you are still getting this warning, you most likely misspelled the timezone identifier. We selected 'America/New_York' for 'EST/-5.0/no DST' instead in **/Users/larryullman/Sites/phpvqs4/templates/header.html** on line **38**

图8-14　对于PHP 5.1，当没有设置时区就调用日期或时间函数时，会生成一个通知

为了演示`date()`函数，我们来修改一下页脚文件，便能显示当前的日期和时间（参见图8-15）。

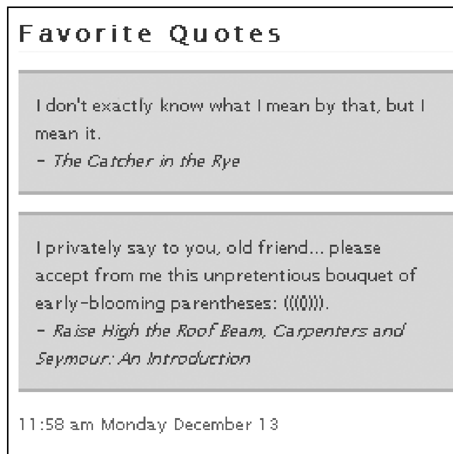


图8-15　使用`date()`函数，Web站点现在可以在边栏中显示日期和时间

⇒ 使用`date()`

(1) 在文本编辑器或IDE中打开`header.html`（参看脚本8-6）。

(2) 在`</div>`关标签之前，添加下面的代码（参看脚本8-7）：

```
<p><?php
```

脚本8-7　修改后的`header.html`文件使用`date()`函数来打印当前日期和时间

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/
    DTD/xhtml11.dtd">
2   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
3   <head>
4     <title><?php // 打印页面标题。
5     if (defined('TITLE')) { // 检查标题是否已定义。
6       print TITLE;
7     } else { // 标题没有定义。
8       print 'Raise High the Roof Beam! A J.D. Salinger Fan Club';
9     }
10    ?></title>
11    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
12    <link rel="stylesheet" href="css/1.css" type="text/css" media="screen,
      projection" />
13  </head>
14  <body>
15  <div id="wrapper">
16
17    <div id="header">
18      <p class="description">A J.D. Salinger Fan Club</p>
19      <h1><a href="index.php">Raise High the Roof Beam!</a></h1>
20      <ul id="nav">
21        <li><a href="books.php">Books</a></li>
22        <li><a href="#">Stories</a></li>
23        <li><a href="#">Quotes</a></li>
24        <li><a href="login.php">Login</a></li>
25        <li><a href="register.php">Register</a></li>
26      </ul>
27    </div><!-- 页头 -->
28
29    <div id="sidebar">
30      <h2>Favorite Quotes</h2>
31        <p class="news">I don't exactly know what I mean by that, but I mean it.
          <br/>- <em>The Catcher in the Rye</em></p>
32        <p class="news">I privately say to you, old friend... please accept from me
          this unpretentious bouquet of early-blooming parentheses: (((()))).<br/>-
          <em>Raise High the Roof Beam, Carpenters and Seymour: An Introduction
          </em></p>
33        <p><?php // 打印当前日期和时间……
34        // 设置时区：
35        date_default_timezone_set('America/New_York');
36
```

```
37         // 打印日期和时间:
38         print date('g:i a l F j');
39         ?></p>
40     </div><!-- 边栏 -->
41
42     <div id="content">
43       <!-- 可变内容开始。-->
44       <!-- 脚本8-7 - header.html -->
```

第一个HTML标签对日期和时间稍微进行了一下格式化，并将其设置为斜体［使用“强调”（em）标签］。然后打开一个PHP节，这样才能调用`date()`函数。

(3) 建立时区：

```
date_default_timezone_set('America/New_York');
```

在调用`date()`之前，必须设置时区。要查找时区，请参见www.php.net/timezones。

(4) 使用`date()`函数来打印当天的日期和时间：

```
print date('g:i a l F j');
```

使用表8-1中列出的格式化参数，这里的`date()`函数将返回一个类似`4:15 pm Tuesday February 22`这样的值。这个值将会被立即打印出来。

(5) 关闭PHP节，完成HTML代码：

```
?></p>
```

(6) 将文件保存为`header.html`，放置在启用了PHP的服务器上的`templates`目录中，并在Web浏览器中进行测试（参见图8-15）。

✓提示

- ❑ 由于PHP是一种服务器端的技术，因此这些函数反映的是服务器上的日期和时间。要获取客户端（换句话说，就是用于浏览页面的Web浏览器所在的计算机）的时间，则必须使用JavaScript。
- ❑ 服务器的时区也可以在PHP配置文件中设置（参见附录A）。在这里建立时区要优于分别在每个脚本中做这项工作。
- ❑ PHP 5.3中加入了一个新的创建和操作日期和时间的方式——`DateTime`类。虽然很有用，但这个新工具需要使用者熟悉面向对象编程，这超出了基础教程的范围，这里不作过多介绍。

8.5　再谈使用 PHP 处理 HTML 表单

到目前为止，本书中所有的示例都是用两个分离的脚本来处理HTML表单：一个用于显示表单，另一个用于接收和处理表单数据。这种方法并没有什么不对，但在一个脚本中完成整个流程则更为有益。为了让一个页面同时显示和处理表单，可以使用一个条件语句（参见图8-16）：

```
if (/* form has been submitted */) {
```

```
        // 处理表单。
} else {
        // 显示表单。
}
```



图8-16 这个流程图说明如何在一个PHP脚本中显示和处理HTML表单

有很多种方式可以检测表单是否已经被提交，一种方法是检查是否为某些变量设置了值：

```
if (isset($_POST['something'])) { ...
```

然而，如果用户在没有完成表单的情况下就进行了提交，则可能没有设置变量的值（取决于对应表单元素的类型）。我的解决方法是在表单中加入一个隐藏文本框来进行检测，这种方法更可靠：

```
<input type="hidden" name="submitted" value="true" />
```

该隐藏输入框的唯一目的就是可靠地指出表单已经被提交。为了检测这一点，用于处理表单的PHP代码可以使用下面的条件语句：

```
if (isset($_POST['submitted'])) { ...
```

另一种检查表单提交情况的方法是检查页面的访问方式。当有一个表单，在提交后返回相同的页面时，脚本会由两种不同类型的请求组成（参见图8-17）。第一个请求是加载表单的GET请求。这是大多数Web页面都会有的标准请求。当表单被提交时，它的`action`属性指向同一个页面，这时会产生第二个请求，此时是POST请求（假设表单使用了POST方法）。因此，可以通过检查表单的请求类型来测试表单的提交情况，使用`$_SERVER`数组：

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') { ...
```

作为示例，我们在这里将建立一个基本的登录表单。

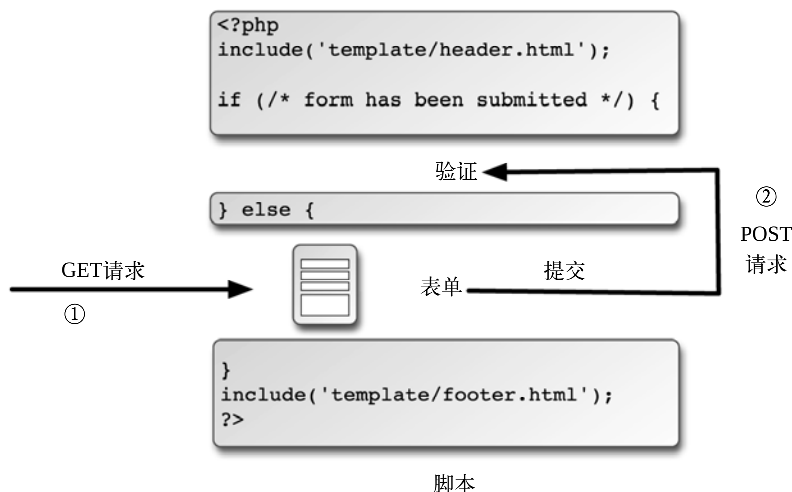


图8-17　当在一个PHP脚本中同时显示和处理HTML表单时，有两种请求脚本的方式

⇒ 使用一个页面来显示和处理表单

(1) 在文本编辑器或IDE中打开一个新的PHP文档，命名为`login.php`（参看脚本8-8）：

```
<?php // 脚本8-8 - login.php
```

脚本8-8　这个登录页用于两种目的。它会显示登录表单并处理其提交的文档

```
1   <?php // 脚本8-8 - login.php
2   /* 页面可以用于用户登录（理论上）。*/
3
4   // 设置页面标题并包含页头文件:
5   define('TITLE', 'Login');
6   include('templates/header.html');
7
8   // 打印一些介绍文本:
9   print '<h2>Login Form</h2>
10    <p>Users who are logged in can take advantage of certain features like this,
      that, and the other thing.</p>';
11
12  // 检查表单是否已提交:
13  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
14
15    // 处理表单:
16    if ( (!empty($_POST['email'])) && (!empty($_POST['password'])) ) {
17
18      if ( (strtolower($_POST['email']) == 'me@example.com')
        && ($_POST['password'] == 'testpass') ) { // 相等!
19
20        print '<p>You are logged in!<br/>Now you can blah,
          blah, blah...</p>';
21
```

```
22        } else { // 不相等!
23
24          print '<p>The submitted email address and password
            do not match those on file!<br/>Go back and try
            again.</p>';
25
26        }
27
28      } else { // 表单未填完整。
29
30        print '<p>Please make sure you enter both an email address
          and a password!<br/>Go back and try again.</p>';
31
32      }
33
34    } else { // 显示表单。
35
36      print '<form action="login.php" method="post">
37      <p>Email Address: <input type="text" name="email" size="20" /></p>
38      <p>Password: <input type="password" name="password"
        size="20" /></p>
39      <p><input type="submit" name="submit" value="Log In!" /></p>
40      </form>';
41
42    }
43
44    include('templates/footer.html'); // Need the footer.
45    ?>
```

8

(2) 将页面标题定义为一个常量，并包含页头文件：

```
define('TITLE', 'Login');
include('templates/header.html');
```

可以使用本章前面介绍的常量系统，为页面设置自己的唯一页面标题。

(3) 添加一些介绍性文本：

```
print '<h2>Login Form</h2>
  <p>Users who are logged in can take advantage of certain
  →features like this, that, and the other thing.</p>';
```

这些文本位于主条件语句之外，不论表单是正在被显示还是已经被提交，它会一直显示在Web浏览器中。因为这个脚本的核心被包含在一个PHP条件语句中，所以需要用PHP打印出HTML表单，而不是像前两个示例（`index.php`和`books.php`）那样将其放在PHP代码之外。

(4) 开启一个条件语句，检测表单是否已经被提交：

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

要测试表单已经被提交的情况，需要检查`$_POST['submitted']`变量是否等于`POST`（区分大小写）。

(5) 创建一对嵌套的条件语句，处理表单数据：

```
if ( (!empty($_POST['email'])) && (!empty($_POST['password'])) ) {
```

```
    if ( (strtolower($_POST['email']) == 'me@example.com') &&
    →($_POST['password'] == 'testpass') ) {
      print '<p>You are logged in!<br/>Now you can blah,
      →blah, blah...</p>';
    } else { // 不相等!
      print '<p>The submitted email address and password do not
      →match those on file!<br/> Go back and try again.</p>';
    }
  } else {
    print '<p>Please make sure you enter both an email address
    →and a password!<br/>Go back and try again.</p>';
  }
```

这些条件语句用于处理表单数据。第一个条件语句检查了Email地址和密码变量是否具有值。如果没有，会显示一个消息（Please make sure…）。在第一个条件语句中，另一个条件语句检查了Email地址是否等于me@example.com以及密码是否等于testpass。如果相等，则认为用户已登录（现在介绍如何存储和检索用户信息有点超前）；否则，会显示一条消息，表示输入了错误的值。要确保在这里的条件语句中使用的是相等运算符（==）而不是赋值运算符（=），这是一种常见的错误。另外，即便用户将Email地址输入成Me@example.com或其他大小写组合方式也没有关系，因为这里在检查相等性之前，首先对Email地址应用了strtolower()函数。

(6) 完成主条件语句：

```
  } else { // 显示表单。
    print '<form action="login.php" method="post">
    <p>Email Address: <input type="text" name="email" size="20" /></p>
    <p>Password: <input type= "password" name="password" size="20" /></p>
    <p><input type="submit" name= "submit" value="Log In!" /></p>
    </form>';

  }
```

这段代码结束了主条件语句，即检查表单是否已经被提交的条件语句。如果未被提交，则会显示表单。这个表单本身非常简单（参见图8-18）。

Login Form

Users who are logged in can take advantage of certain features like this, that, and the other thing.

Email Address:

Password:

Log In!

图8-18　这个简单的登录页接受一个Email地址和一个密码

为了避免混淆，即使表单`method`属性有一个值为post（全部小写），在检查表单提交情况时，请求方法的值仍应为POST（全部大写）。

(7) 请求页脚文件并完成PHP页面：

```
include('templates/footer.html');
?>
```

(8) 将文件保存为`login.php`，放置在和`index.php`相同的目录中，然后在Web浏览器中进行测试（参见图8-19、图8-20和图8-21）。

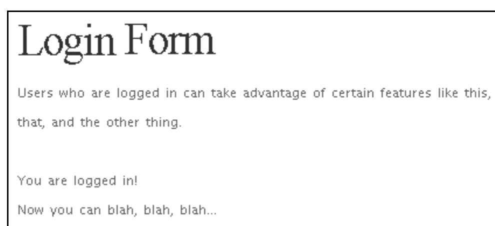


图8-19　成功登录后，用户会看到这样的消息

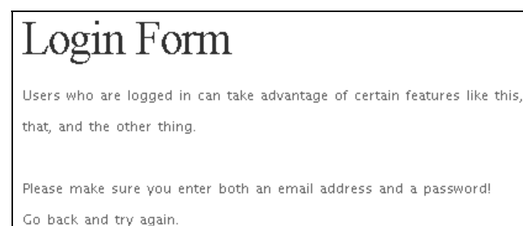


图8-20　如果没有提交Email地址或密码，则会看到这样的消息



图8-21　如果与脚本中的Email地址或密码不匹配，则用户会看到这样的错误消息

✓提示

- 在实际使用当中，可能需要为错误消息添加一些CSS样式，使其更加突出。本章的8.6节将介绍这一功能。
- 这里用于判断输入框是否存在的小技巧可能让人迷惑。它的工作原理是因为同样的脚本——`login.php`——会被用户访问两次。第一次访问时表单并没有被提交，所以主条件会是FALSE，然后显示表单。在用户单击submit之后，会再次访问页面，此时条件会变为TRUE。
- 如果希望页面在处理完表单之后，能够立即再次显示表单，可以这样做：

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
  // 处理表单。
}
// 显示表单。
```

8.6　使表单更具粘性

粘性表单（sticky form）会记住输入到其中的值。一个常见的示例就是搜索引擎，它总会在搜索框中显示所输入的关键词，甚至在显示搜索结果时亦是如此。当用户由于输入错误值而没能准确完成表单，从而需要重新提交时，可能就需要使用粘性表单了（参见图8-22）。

Registration Form

Register so that you can take advantage of certain features like this, that, and the other thing.

Please enter your last name!

Please enter your email address!

Please enter a password!

Please try again!

First Name: Larry

Last Name:

图8-22　在没有完成表单提交时，粘性表单可以帮用户找到漏填的表单值

从技术的角度来看，粘性表单可以通过为表单元素提供预设值来实现。可以在创建表单时为输入框设置value输入框：

```
<input type="text" name=
→"first_name" value="Stephanie" />
```

要使用PHP预设值，可以在引号中打印适当的变量：

```
<input type="text" name="first_name"
→value="<?php print $_POST ['first_name']; ?>" />
```

表单第一次运行时，PHP代码什么都不打印（因为变量没有值）。如果表单在提交后再次显示，则用户之前在表单中输入的值会自动显示出来。这是基本的方法，但更加专业的实现方式需要处理两件事……

首先，最好不要引用不存在的变量。这样会引发PHP警告，而如果将PHP代码嵌入到一个表单元素属性中，警告代码只能出现在HTML的源代码中（参见图8-23）。为了避免这一点，需要在打印之前检查变量是否已被设置。

```
<input type="text" name="first_name" value="<?php if (isset($_POST
→['first_name'])) { print $_POST['first_name']; } ?>" />
```

```
        <p>First Name: <input type="text" name="first_name" size="20" value="<br />
<b>Notice</b>:  Undefined index: first_name in <b>/Users/larryullman/Sites/phpvqs4/register.php</b>
" /></p>
```

图8-23　HTML源代码，显示由于引用不存在变量而产生的PHP页面错误

其次，用户提交的值中的某些字符，直接打印在表单元素的value属性中可能会引发问题。要避免这样的问题，需要使用htmlspecialchars()函数（在第5章中有所讨论）。考虑到这一点，下面给出了这段代码更长但却是更好的版本：

```
<input type="text" name="first_name" value="<?php if (isset($_POST
→['first_name'])) { print htmlspecialchars($_POST['first_name']); } ?>" />
```

作为演示，我们来创建注册表单的界面（参见图8-24）。

图8-24　用户首次看到的注册表单

⇛ 使表单更具粘性的步骤

(1) 在文本编辑器或IDE中创建一个新的PHP脚本，命名为`register.php`（参看脚本8-9）：

```
<?php // 脚本8-9 - register.php
```

脚本8-9　注册表单使用了所谓的粘性特征，这样它就可以回显用户输入的值

```
1   <?php // 脚本8-9 - register.php
2   /* 页面可以用于用户登录（理论上）。*/
3
4   // 设置页面标题并包含头文件：
5   define('TITLE', 'Register');
6   include('templates/header.html');
7
8   // 打印一些介绍文本：
9   print '<h2>Registration Form</h2>
10    <p>Register so that you can take advantage of certain features like this, that,
      and the other thing.</p>';
11
12  // 添加CSS：
13  print '<style type="text/css" media="screen">
14    .error { color: red; }
15  </style>';
16
17  // 检查表单是否已提交：
18  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
19
20    $problem = FALSE; // 目前一切正常。
21
22    // 检查每个值……
23    if (empty($_POST['first_name'])) {
24      $problem = TRUE;
25      print '<p class="error">Please enter your first name!</p>';
26    }
```

```
27
28    if (empty($_POST['last_name'])) {
29      $problem = TRUE;
30      print '<p class="error">Please enter your last name!</p>';
31    }
32
33    if (empty($_POST['email'])) {
34      $problem = TRUE;
35      print '<p class="error">Please enter your email address!</p>';
36    }
37
38    if (empty($_POST['password1'])) {
39      $problem = TRUE;
40      print '<p class="error">Please enter a password!</p>';
41    }
42
43    if ($_POST['password1'] != $_POST['password2']) {
44      $problem = TRUE;
45      print '<p class="error">Your password did not match your confirmed
        password!</p>';
46    }
47
48    if (!$problem) { // 判断是否有错误发生……
49
50      // 打印一条消息：
51      print '<p>You are now registered!
        <br/>Okay, you are not really registered but...</p>';
52
53      // 清除数组中的元素：
54      $_POST = array();
55
56    } else { // 表单未填完整。
57
58      print '<p class="error">Please try again!</p>';
59
60    }
61
62  } // 结束处理表单的条件语句。
63
64  // 创建表单：
65  ?>
66  <form action="register.php" method="post">
67
68    <p>First Name: <input type="text" name="first_name" size="20"
      value="<?php if (isset($_POST ['first_name'])) { print
      htmlspecialchars($_POST['first_name']); } ?>" /></p>
69
70    <p>Last Name: <input type="text" name="last_name" size="20"
      value="<?php if (isset($_POST['last_name'])) { print
      htmlspecialchars($_POST['last_name']); } ?>" /></p>
71
72    <p>Email Address: <input type="text" name="email"
      size="20" value="<?php if (isset($_POST['email'])) { print
      htmlspecialchars($_POST['email']); } ?>" /></p>
```

```
73
74      <p>Password: <input type="password" name="password1"
        size="20" value="<?php if (isset($_POST['password1']))
        { print htmlspecialchars($_POST['password1']); } ?>" /></p>
75      <p>Confirm Password: <input type="password" name="password2"
        size="20" value="<?php if (isset($_POST['password2']))
        { print htmlspecialchars($_POST['password2']); } ?>" /></p>
76
77      <p><input type="submit" name="submit" value="Register!" /></p>
78
79    </form>
80
81    <?php include('templates/footer.html'); // Need the footer. ?>
```

(2) 设置页面标题并包含HTML页头：

```
define('TITLE', 'Register');
include('templates/header.html');
```

(3) 添加一些介绍性文本并定义一个CSS类：

```
print '<h2>Registration Form</h2>
  <p>Register so that you can take advantage of certain
  →features like this, that, and the other thing.</p>';
print '<style type="text/css" media="screen">
  .error { color: red; }
</style>';
```

如果没能正确完成注册表单，为了更加突出生成的错误消息，这里定义了一个CSS类，它会使文字的颜色变为红色。尽管CSS通常定义在页面的顶部，但实际上可以将其放在任何位置。

(4) 检查表单是否已经被提交：

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

和登录页一样，该脚本同时显示和处理注册表单。检查表单是否已经提交，这里使用的代码与前面示例中的一样。

(5) 创建一个标志变量：

```
$problem = FALSE;
```

`$problem`变量用于指出是否出现错误。在正式注册用户之前，需要确保每个表单输入框都已完全填写。在一开始，该变量被设置为FALSE，因为尚无错误发生。

(6) 检查输入的`first_name`：

```
if (empty($_POST['first_name'])) {
  $problem = TRUE;
  print '<p class="error">Please enter your first name!</p>';
}
```

检查用户是否输入了`first_name`值是一个非常简单的测试，只需判断该变量是否为非空。（该技术在第6章中首次介绍。）如果该变量是空，则通过将标志变量设置为TRUE来指出错误，并打印一条错误消息。错误消息的`class`值是`error`，因此会向其应用CSS样式。

(7) 对last_name和Email地址重复这个验证：

```
if (empty($_POST['last_name'])) {
  $problem = TRUE;
  print '<p class="error">Please enter your last name!</p>';
}
if (empty($_POST['email'])) {
  $problem = TRUE;
  print '<p class="error">Please enter your email address!</p>';
}
```

这两个检查是对用户名的检查代码进行修改后得到的。

(8) 验证密码：

```
if (empty($_POST['password1'])) {
  $problem = TRUE;
  print '<p class="error">Please enter a password!</p>';
}
if ($_POST['password1'] != $_POST['password2']) {
  $problem = TRUE;
  print '<p class="error">Your password did not match your confirmed password!</p>';
}
```

密码的验证需要两个条件语句。第一个条件检查了$_POST['password1']变量是否为空。第二个条件检查了$_POST['password1']变量是否等于$_POST['password2']变量。这里无需检查$_POST['password2']是否为空，因为如果它为空而$_POST['password1']不为空，则第二个条件会捕获这个问题。如果$_POST['password2']和$_POST['password1']同时为空，则第一个条件会捕获这种问题。

(9) 检查是否有错误发生：

```
if (!$problem) {
  print '<p>You are now registered!<br/>Okay, you are not really registered but...</p>';
  $_POST = array();
```

如果没有错误，则$problem变量仍然为FALSE，该代码第一行中的条件将为TRUE（这是因为$problem的值为FALSE）。在这种情况下，注册流程将得以执行。正式的注册流程需要将数据存放在文件或数据库中，这里还没有开发这些代码，所以只是用一个简单的消息来替代。

接下来，用array()为$_POST变量赋值。这一行代码的效果是擦除了$_POST变量中的内容（也就是说，将其重置为一个空数组）。只在注册成功（理论上）之后执行这一步，这样输入的值就不会再次显示在注册表单中（参见第(12)步）。

(10) 完成条件语句：

```
  } else { // 表单未填完整。
    print '<p class="error">Please try again!</p>';
  }
} // 结束处理表单的条件语句。
```

如果发生了错误则会使用else子句，要求用户重新完成表单。

(11) 打开HTML表单：

```
?>
<form action="register.php" method="post">
```

和登录示例不同，该页面总是显示表单。因此，表单并不属于条件语句的一部分。另外，由于要生成大量的HTML，所以暂时离开PHP节，直接输出HTML会比较方便一些。

(12) 创建粘性的first name输入框：

```
<p>First Name: <input type="text" name="first_name" size="20"
→value="<?php if (isset($_POST['first_name'])) { print
→htmlspecialchars($_POST['first_name']); } ?>" /></p>
```

为了让first_name输入具有粘性,需要通过打印$_POST['first_name']变量的值来预设其value属性，但只在已设置该变量的值时这样做。因此，需要将这个条件放到HTML表单元素value节中的PHP标签里。如前所述，需要使用htmlspecialchars()函数来处理可能存在问题的字符。

注意，如果用户正确填写完表单，$_POST数组会被重置，使PHP的条件语句为FALSE。

(13) 针对last_name和Email地址重复这一过程：

```
<p>Last Name: <input type="text" name="last_name" size="20"
→value="<?php if (isset($_POST['last_name'])) { print
→htmlspecialchars($_POST['last_name']); } ?>" /></p>
<p>Email Address: <input
→type="text" name="email" size="20" value="<?php if
→(isset($_POST['email'])) { print htmlspecialchars($_POST
→['email']); } ?>" /></p>
```

这些代码是从第(12)步中演变过来的，适当地修改了变量名。

(14) 添加表单的剩余部分：

```
<p>Password: <input type="password" name="password1" size="20" value="<?php if
→(isset($_POST['password1'])) { print htmlspecialchars
→($_POST['password1']); } ?>" /></p>
  <p>Confirm Password: <input type="password" name="password2"
  →size="20" value="<?php if (isset($_POST['password2']))
  →{ print htmlspecialchars($_POST['password2']); }
  →?>" /></p>
<p><input type="submit" name="submit" value="Register!" /></p>
</form>
```

以前不能为password类型的输入框设置预设值，但现在一些浏览器已支持这个功能。接下来放置了提交按钮和form关闭标签。

(15) 完成PHP页面：

```
<?php include('templates/footer.html'); ?>
```

最后一步是包含HTML页脚文件。

(16) 将文件保存为register.php，放置在启用了PHP的服务器上的适当目录中，并在Web浏览器中进行测试（参见图8-25和图8-26）。

图8-25　注册表单指出了发生的错误，并保留了除密码之外的表单值

图8-26　用户填写成功之后的注册表单

✓提示

- 根据(X)HTML标准，必须将表单输入框的属性放在引号中，特别是要放在双引号中。如果没有用引号把值引起来，则其中任何一个空格都会标志着值的结束（例如，Larry Ullman只会在表单输入框中显示出Larry）。虽然在HTML5中，输入框的属性不必再用引号括起来，但我建议你加上引号。
- 要为单选按钮或复选框设置预设值，应该在其文本框标签中添加`checked="checked"`：

  ```
  <input type="checkbox" name="interests[ ]" value="Skiing" checked="checked" />
  ```

  当然，需要使用一个PHP条件语句，来确定是否应该将这些文字添加到元素的定义中。
- 要为下拉框设置预选值，请使用`selected="selected"`：

  ```
  <select name="year">
  <option value="2011">2011</option>
  <option value="2012" selected="selected">2012</option>
  </select>
  ```

  同样，需要使用一个PHP条件语句来检查是否需要将这些文字添加到元素定义中。
- 要为文本区设置预设值，需要将值放置在textarea标签之内：

  ```
  <textarea name="comments" rows="10" cols="50">preset value</textarea>
  ```

8.7 发送 Email

从理论上说，使用PHP发送Email非常简单，只需要使用`mail()`函数即可。`mail()`函数使用服务器上的Email应用程序（如Unix或Mac OS X上的`sendmail`）或SMTP（Simple Mail Transfer Protocol）服务器来向外发送消息。该函数的基本使用方法是：

```
mail(to, subject, body);
```

第一个参数是Email应该被发送的地址（可以是多个地址，使用逗号分隔）。第二个参数是消息的主题，第三个参数是消息的内容。

该函数还可以接受另外一个参数，用于为Email添加更多详细信息（附加头），包括发件人地址、Email优先级和抄送地址：

```
mail('someone@example.com', 'Test Email', 'This is a test email', 'From:
→'email@example.com');
```

尽管这从理论上说很容易，但在显示代码中实际使用该函数时会更为复杂。对于初学者来说，设置自己的计算机来发送Email是很难的（请参见框注“配置服务器以发送Email”）。

其次，需要执行一些步骤来防止恶意用户试图用表单来发送垃圾邮件。在下一个示例中，会将一封Email发送到用户提交的Email地址。这里允许用户提交多个Email地址（参见图8-27），邮件会分发到每个Email地址。有很多种安全防护措施。本书采用一种相对简单的方法，即确认提供的一个Email地址中只包含一个“@”标志。可以使用`substr_count()`函数计算字符串中子字符串的数量：

```
if (substr_count($_POST['email'],'@') == 1) {...
```

Email Address:
me@example.com,you@example.edu,whomever@example.net

图8-27　用户可以使用类似的表单将邮件发送给多个收件人

配置服务器以发送Email

只要将Web服务器（或计算机）设置为可以发送Email，则使用PHP发送Email还是很方便的。如果使用了Web主机服务，或自己的Unix计算机（类似于Linux），那么这并不成问题。如果有问题，请联系托管公司寻求支持。

如果你使用自己的服务器（在本地开发），那么发送Email是关键功能。如果你使用的是多合一安装包（如MAMP或XAMPP，参见附录A），那么安装包中还会包含Email服务。如果在注册后没有收到邮件，查看相关的软件说明文档，了解如何启用Email服务。

如果你使用操作系统内置的Web服务器软件（如在Mac OS X运行的Apache），可能已经配置（也可能没有配置）Email发送服务。一开始可以使用一个有效的Email地址继续尝试这个示例。如果没有收到Email，请参见附录A中的更多信息，以让`mail()`能够工作。

要补充的是，我几乎从没担心过能不能用PHP在我的计算机上发送Email，因为我从不在自己的计算机上运行实际的Web站点。换句话说，为什么要把时间花在那些最终并不会使用的地方呢（应该担心的是使用PHP在实际的服务器上发送Email）。

下面来向注册页面中添加一个对`mail()`函数的调用，这样你就能对该函数的使用方式有一个感性的认识。

⇒ 使用PHP发送邮件

(1) 在文本编辑器或IDE中打开`register.php`（参看脚本8-9）。

(2) 修改Email验证，检查Email地址中是否只有一个“@”标志（参看脚本8-10）：

```
if (empty($_POST['email']) || (substr_count($_POST['email'],'@') != 1) ) {
```

如果地址为空或“@”标志的个数不是1的话，就不能通过Email验证。这个验证并不完美（还差得很远），但是这已经让使用Email地址的安全风险变得小多了。参见本节后面的提示，改进这个验证。

脚本8-10　在PHP中，可以通过`mail()`函数来发送Email

```
1   <?php // 脚本8-10 - register.php #2
2   /* 页面可以用于用户登录（理论上）。*/
3
4   // 设置页面标题并包含头文件：
5   define('TITLE', 'Register');
6   include('templates/header.html');
7
8   // 打印一些介绍文本：
9   print '<h2>Registration Form</h2>
10    <p>Register so that you can take advantage of certain features like this,
      that, and the other thing.</p>';
11
12  // 添加CSS：
13  print '<style type="text/css" media="screen">
14    .error { color: red; }
15  </style>';
16
17  // 检查表单是否已提交：
18  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
19
20    $problem = FALSE; // 目前一切正常。
21
22    // 检查每个值……
23    if (empty($_POST['first_name'])) {
24      $problem = TRUE;
25      print '<p class="error">Please enter your first name!</p>';
26    }
27
28    if (empty($_POST['last_name'])) {
29      $problem = TRUE;
```

```
30       print '<p class="error">Please enter your last name!</p>';
31     }
32
33     if (empty($_POST['email']) || (substr_count($_POST['email'], '@') != 1) ) {
34       $problem = TRUE;
35       print '<p class="error">Please enter your email address!</p>';
36     }
37
38     if (empty($_POST['password1'])) {
39       $problem = TRUE;
40       print '<p class="error">Please enter a password!</p>';
41     }
42
43     if ($_POST['password1'] != $_POST['password2']) {
44       $problem = TRUE;
45       print '<p class="error">Your password did not match your confirmed
         password!</p>';
46     }
47
48     if (!$problem) { // 判断是否有错误发生……
49
50       // 打印一条消息:
51       print '<p>You are now registered!<br/>Okay, you are not really registered
         but...</p>';
52
53       // 发送Email:
54       $body = "Thank you for registering with the J.D. Salinger fan club! Your password
         is '{$_POST['password1']}'.";
55       mail($_POST['email'], 'Registration Confirmation', $body, 'From:
         admin@example.com');
56
57       // 清除数组中的元素:
58       $_POST = array();
59
60     } else { // 表单未填完整。
61
62       print '<p class="error">Please try again!</p>';
63
64     }
65
66   } // 结束处理表单的条件语句。
67
68   // 创建表单:
69   ?>
70   <form action="register.php" method="post">
71
72     <p>First Name: <input type="text" name="first_name" size="20" value="<?php if
         (isset($_POST['first_name'])) { print htmlspecialchars($_POST['first
         _name']); } ?>" /></p>
73
74     <p>Last Name: <input type="text" name="last_name" size="20" value="<?php if
     (isset($_POST['last_name'])) { print htmlspecialchars($_POST['last_name']);
     } ?>" /></p>
75
```

8

```
76    <p>Email Address: <input type="text" name="email" size="20" value="<?php if
      (isset($_POST['email'])) { print htmlspecialchars($_POST['email']); } ?>" />
      </p>
77
78    <p>Password: <input type="password" name="password1" size="20" value="<?php if
      (isset($_POST['password1'])) { print htmlspecialchars($_POST['password1']); }
      ?>" /></p>
79    <p>Confirm Password: <input type="password" name="password2" size="20" value=
      "<?php if (isset($_POST['password2'])) { print htmlspecialchars($_POST['password2']);
      } ?>" /></p>
80
81    <p><input type="submit" name="submit" value="Register!" /></p>
82
83  </form>
84
85  <?php include('templates/footer.html'); // 包含页脚文件。?>
```

(3) 在注册消息之后（参见第51行），添加下面的代码：

```
$body = "Thank you for registering with the J.D. Salinger fan
→club! Your password is '{$_POST['password1']}'.";
mail($_POST['email'], 'Registration Confirmation', $body, 'From:
→admin@example.com');
```

有时使用该函数最简单的方式，是将正文建立在一个变量中，然后传递给`mail()`函数。消息本身将被发送到用户注册时提供的地址中，标题为Registration Confirmation，来自admin@example.com。如果在一台真实的服务器上运行该程序，应该在FROM值中使用该站点的真实Email地址。

(4) 保存文件，将其放置在支持PHP和Email的服务器上的适当目录中，并在Web浏览器中进行测试（参见图8-28）。

图8-28　再次测试注册表单

(5) 填写完表单，检查邮箱中的消息（参见图8-29）。



图8-29　这是当伪注册过程成功后，由PHP脚本发送的Email

✓**提示**

- 8.10节会介绍一个非常好的验证Email地址的工具，前提是PHP必须是5.2及以上版本。
- 在我编写的《PHP 6与MySQL 5基础教程》[①]一书以及本书的在线论坛（www.LarryUllman.com/forum/）中，讨论了通过PHP脚本确保Email发送成功的其他方法。
- 如果在接收由PHP发送的Email时遇到问题，首先请确保在不使用PHP的情况下，邮件服务器能够正常工作。然后确认使用了有效的FROM地址。最后，尝试使用一个不同的接收地址，并查看一下垃圾邮件，看看消息是否被放到这里了（如果可以的话）。
- 还可以发送带有附件的Email或HTML格式的Email，但这样做需要更多复杂的编码（通常会引入类和对象）。幸运的是，很多程序员已经开发了可以使用的解决方案。请参见附录B，其中给出的Web站点可能会有所帮助。
- `mail()`函数会返回一个值（1或0）表示它是否执行成功。这个值只能表示PHP是否能够尝试发送Email（通过判断是否安装有Email系统）。没有哪种简单的方法能够使用PHP判断一个Email地址是否有效，或最终用户是否收到了消息。
- 要向多个地址发送一封Email，可以使用CC参数，或者用逗号分隔TO参数中的地址。
- 要在Email正文中创建新行，可以在创建消息时使用多行文本，或者在双引号中使用换行符（`\n`）。
- 如果除了FROM地址之外，还想发送多个头，可以使用`\r\n`组合将它们分开：

```
mail ('email@example.com','Testing', $body, "From:email@example.org
→\r\nBcc:hidden@example.net,third@example.com");
```

8.8　输出缓冲

本章和后面章节中用到的某些函数，只能在没有任何东西被发送到浏览器之前调用。这些函数包括`header()`、`setcookie()`和`session_start()`。如果在Web浏览器已经收到了一些文本、HTML或哪怕是一个空格之后调用这些函数，就会得到一个恼人的HTTP头已发送错误消息（参见图8-30）。

① 本书简体中文版由人民邮电出版社于2007年出版。——编者注

```
Warning: Cannot modify header information - headers already sent by (output started
at /Users/larryullman/Sites/phpvqs4/templates/header.html:4) in /Users/larryullman
/Sites/phpvqs4/login.php on line 22
```

图8-30　如果在调用header()之前浏览器接收到了任何HTML，都会看到这样的错误消息

本书为初学PHP的开发者提供了一种解决方法，即采用*输出缓冲*（output buffering，或称输出控制）。在一般的PHP脚本中，PHP标签之外的任何HTML都会立即发送到Web浏览器，一旦执行了print语句，所有的打印内容也是如此。利用输出缓冲，HTML和打印的数据（输出）将被放到缓冲（也就是内存）中。当脚本执行结束后，缓冲将被发送到Web浏览器，或者如果需要的话，缓冲可以清空而不发送到Web浏览器。使用输出缓冲的原因有很多，但对于初学者来说，一个主要的优点在于，使用这些函数时，无需担心会出现“HTTP头已发送”的错误。尽管还没有用到这些函数，但本章还是在这里简要介绍了输出缓冲，因为它极大程度地减少了在使用HTTP头（8.9节介绍）、cookie（第9章介绍）和session（第9章介绍）时的错误。

要启用输出缓冲，请在页面的最顶端使用ob_start()函数。一但调用了该函数，则每个print和类似的函数都会将数据发送到内存缓冲中，而不是发送到Web浏览器。相反，HTTP调用（如header()和setcookie()）不会缓冲，而是按照常规方式处理。

在脚本的结尾处，调用ob_end_flush()函数，将积累下来的缓冲发送到Web浏览器。或者，可以使用ob_end_clean()函数删除缓冲的数据而不进行传输。这两个函数都会为当前脚本关闭输出缓冲。

从程序员的视角来看，输出缓冲允许以更加线性的方式来组织脚本，而不用关心HTTP头。下面我们来修改一下header.html和footer.html，这样每个页面都能使用输出缓冲。目前你可能不会从中受益，但在以后编写复杂的程序时，它可能会为你减少很多出错的机会，使你保持清醒的头脑。

⇛ 使用输出缓冲

(1) 在文本编辑器或IDE中打开header.html（参看脚本8-6）。

(2) 在页面的最顶端、任何HTML代码之前，添加下面的代码（参看脚本8-11）：

```
<?php
ob_start();
?>
```

脚本8-11　将ob_start()函数放在header.html脚本的最顶端，为Web应用程序添加输出缓冲

```
1    <?php // 脚本8-11 - header.html #4
2
3    // 开启输出缓冲：
4    ob_start();
5
6    ?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
7    <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
8    <head>
```

```
9     <title><?php // 打印页面标题。
10    if (defined('TITLE')) { // 检查标题是否已定义。
11      print TITLE;
12    } else { // 标题没有定义。
13      print 'Raise High the Roof Beam! A J.D. Salinger Fan Club';
14    }
15    ?></title>
16    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
17    <link rel="stylesheet" href="css/1.css" type="text/css"
      media="screen,projection" />
18  </head>
19  <body>
20  <div id="wrapper">
21
22    <div id="header">
23      <p class="description">A J.D. Salinger Fan Club</p>
24      <h1><a href="index.php">Raise High the Roof Beam!</a></h1>
25      <ul id="nav">
26        <li><a href="books.php">Books</a></li>
27        <li><a href="#">Stories</a></li>
28        <li><a href="#">Quotes</a></li>
29        <li><a href="login.php">Login</a></li>
30        <li><a href="register.php">Register</a></li>
31      </ul>
32    </div><!-- 页头 -->
33
34    <div id="sidebar">
35      <h2>Favorite Quotes</h2>
36        <p class="news">I don't exactly know what I mean by that, but I mean it.
          <br/>- <em>The Catcher in the Rye</em></p>
37        <p class="news">I privately say to you, old friend... please accept from me
          this unpretentious bouquet of early-blooming parentheses: (((()))).<br/>-
          <em>Raise High the Roof Beam, Carpenters and Seymour: An Introduction</em>
          </p>
38        <p><?php // 打印当前日期和时间……
39        // 设置时区：
40        date_default_timezone_set('America/New_York');
41
42        // 打印日期和时间：
43        print date('g:i a l F j');
44        ?></p>
45    </div><!-- 边栏 -->
46
47    <div id="content">
48      <!-- 可变内容开始。-->
```

使用输出缓冲的关键在于，要在脚本中尽可能早地调用`ob_start()`函数。这个示例中，在任何HTML之前创建了一个特殊的PHP节，并在这里调用了`ob_start()`。在页头文件中打开输出缓冲，并在页脚文件中将其关闭，这样做可以让Web应用程序中的每个页面都能使用输出缓冲。

(3) 在文本编辑器或IDE中打开`footer.html`（参看脚本8-8）。

(4) 在脚本的结束位置，所有的HTML之后，添加（参看脚本8-12）：

8

```
<?php
ob_end_flush();
?>
```

脚本8-12　在页脚文件的结尾处使用ob_end_flush()完成了输出缓冲，这会将积累下来的缓冲发送到Web浏览器

```
1        <!-- 可变内容结束。-->
2      </div><!-- 内容 -->
3
4      <div id="footer">
5        <p>Template design by <a href="http://www.sixshootermedia.com">
         Six Shooter Media</a>.</p>
6        <p>&copy; 2011</p>
7      </div><!-- 页脚 -->
8
9    </div><!-- 完成页面 -->
10   </body>
11   </html><?php // 脚本8-12 - footer.html #2
12
13   // 将缓冲内容发送到浏览器并关闭输出缓冲:
14   ob_end_flush();
15   ?>
```

该代码将积累下来的缓冲内容发送到Web浏览器，并关闭输出缓冲。换句话说，所有的HTML都是在此刻发送的。

(5) 保存这两个文件，并将其放置在启用了PHP的服务器上的templates目录中。

(6) 在Web浏览器中测试任意一个页面（参见图8-31）。

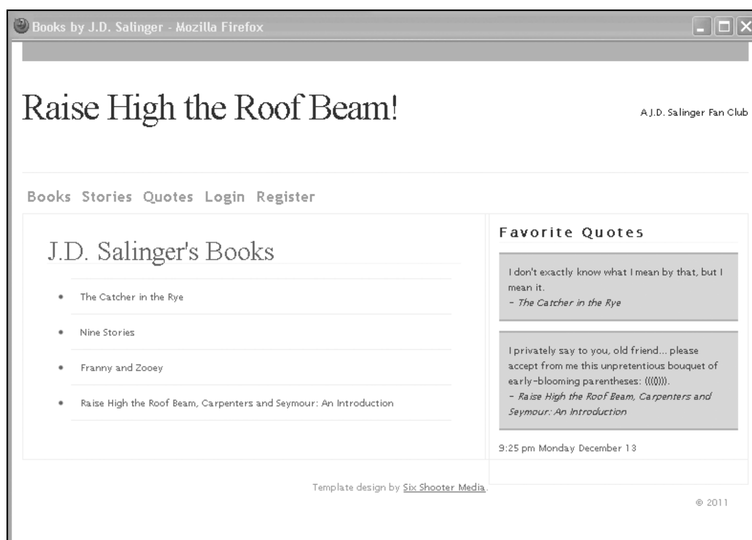


图8-31　站点一如既往地工作，但当在使用本章稍后介绍的HTTP头之后，将变得更加容易

✓提示

- 要澄清的是，如果该文件会被包含到PHP脚本（如index.php）中的话（就像这里这两个示例一样），PHP代码可以放置在有.html扩展名的文件中。
- 现在，PHP的默认配置会自动启用输出缓冲。
- 可以在php.ini文件中设置最大缓冲大小，默认值是4 096字节。
- ob_get_length()函数返回当前缓冲内容的长度（字符数）。
- ob_get_contents()函数可以返回当前缓冲区的内容，如果需要的话可以将其赋值给一个变量。
- ob_flush()函数可以将缓冲中的当前内容发送到Web浏览器中，并丢弃这些内容，以便能够启动新的缓冲。该函数可以使脚本保持更稳定的缓冲大小。
- ob_clean()函数删除缓冲中的当前内容，而不会停止缓冲过程。
- 如果没有调用ob_end_flush()函数，PHP会在脚本结束后自动调用它。但自己调用该函数仍然是一种好习惯。

8.9 处理 HTTP 头

服务器和Web浏览器（客户端）的很多交互都是通过HTTP（Hypertext Transfer Protocol）进行的。这就是Web页面的地址都要以http://开头的原因。但是，除了发送HTML、图片等信息外，Web服务器通常需要用其他一些方式与客户端通信。这些额外的通信需要使用HTTP头实现。HTTP头有很多种用法，所有这些HTTP头都可以使用PHP的header()函数来处理。

本节演示了header()函数的一种常见的用法：将用户从一个页面重定向到另一个页面。要使用PHP来重定向用户的浏览器，需要发送一个location头：

```
header('Location: page.php');
```

通常，该header()函数会后跟一个exit()，用以取消当前脚本的执行（因为浏览器已经被重定向到另外一个页面了）：

```
header('Location: page.php');
exit();
```

使用header()需要了解的最重要的一件事就是，必须在任何内容被发送到Web浏览器之前调用该函数——否则，会看到非常常见的“HTTP头已发送”错误消息（参见图8-30）。如果Web浏览器接收到了任何HTML或哪怕是一个空格，header()函数都无法工作。

幸运的是，8.8节已经介绍了输出缓冲。因为在Web应用程序中打开了输出缓冲，所以任何内容都不会立即发送给Web浏览器，直到页脚脚本的最后一行（当调用ob_end_flush()时）。通过这种方法，可以避免“HTTP头已发送”的错误。

为了演示重定向，下面我们重写登录页，在成功登录后将用户转到欢迎页。

⇒ 使用`header()`函数

(1) 在文本编辑器或IDE中打开`login.php`（参看脚本8-8）。

(2) 删除“You are logged in…”这条`print`语句（参看脚本8-13）。

脚本8-13　新版本的登录页会使用`header()`函数将用户重定向到另一个页面

```
1   <?php // 脚本8-13 - login.php #2
2   /* 页面可以用于用户登录（理论上）。*/
3
4   // 设置页面标题并包含页头文件：
5   define('TITLE', 'Login');
6   include('templates/header.html');
7
8   // 打印一些介绍文本：
9   print '<h2>Login Form</h2>
10    <p>Users who are logged in can take advantage of certain features like this, that,
      and the other thing.</p>';
11
12  // 检查表单是否已提交：
13  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
14
15    // 处理表单：
16    if ( (!empty($_POST['email'])) && (!empty($_POST['password'])) ) {
17
18      if ( (strtolower($_POST['email']) == 'me@example.com') && ($_POST['password']
        == 'testpass') ) { // 相等！
19
20        // 将用户重定向到其他页面！
21        ob_end_clean(); // 清空缓冲！
22        header ('Location:
          welcome.php');
23        exit();
24
25      } else { // 不相等！
26
27        print '<p>The submitted email address and password do not match those on
          file!<br/>Go back and try again.</p>';
28
29      }
30
31    } else { // 表单未填完整。
32
33      print '<p>Please make sure you enter both an email address and a password!<br
        />Go back and try again.</p>';
34
35    }
36
37  } else { // 显示表单。
38
39    print '<form action="login.php" method="post">
40    <p>Email Address: <input type="text" name="email" size="20" /></p>
41    <p>Password: <input type="password" name="password" size="20" /></p>
```

```
42     <p><input type="submit" name="submit" value="Log In!" /></p>
43     </form>';
44
45  }
46
47  include('templates/footer.html'); // 包含页脚文件。
48  ?>
```

由于用户将被重定向到其他页面，因此这里不再需要这条消息了。

(3) 在原来print语句的地方，添加：

```
ob_end_clean( );
header ('Location: welcome.php');
exit( );
```

第一行代码销毁了页面缓冲（因为此时积累下来的缓冲不再有用）。这样做并不是必需的，但的确是个好主意。下一行代码将用户重定向到welcome.php。第三行代码结束了该脚本剩余部分的执行。

(4) 保存该文件，并放到启用了PHP的服务器上的适当目录中（和本章其他脚本放在一起）。

接下来需要创建一个welcome.php页面，用户将被重定向到该页面。

⇒ 编写welcome.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为welcome.php（参看脚本8-14）：

```
<?php // 脚本8-14 - welcome.php
```

脚本8-14 欢迎页会在用户登录之后显示欢迎辞

```
1   <?php // 脚本8-14 - welcome.php
2   /* 这是欢迎页，用户成功登录后
3   会被重定向到这里。*/
4
5   // 设置页面标题并包含头文件：
6   define('TITLE', 'Welcome to the J.D. Salinger Fan Club!');
7   include('templates/header.html');
8
9   // 关闭PHP节创建HTML内容：
10  ?>
11
12  <h2>Welcome to the J.D. Salinger Fan Club!</h2>
13  <p>You've successfully logged in and can now take advantage of everything the site
    has to offer.</p>
14  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
    incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute
    irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
    pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
    deserunt mollit anim id est laborum.</p>
15
16  <?php include('templates/footer.html'); // 包含页脚文件。?>
```


(2) 定义页面的标题，并包含页头文件：

```
define('TITLE', 'Welcome to the J.D. Salinger Fan Club!');
include('templates/header.html');
```

(3) 创建页面内容：

```
?>
<h2>Welcome to the J.D. Salinger Fan Club!</h2>
<p>You've successfully logged in and can now take advantage
→of everything the site has to offer.</p>
```

(4) 返回PHP并包含页脚文件：

```
<?php include('templates/footer.html'); ?>
```

(5) 将脚本保存为`welcome.php`，放置到与新版本的`login.php`相同的目录中，并在Web浏览器中进行测试（参见图8-32、图8-33和图8-34）。

Login Form

Users who are logged in can take advantage of certain features like this, that, and the other thing.

Email Address: me@example.com

Password: ●●●●●●●●●

Log In!

图8-32　登录表单



图8-33　以及当用户成功登录后进行的重定向

Login Form

Users who are logged in can take advantage of certain features like this, that, and the other thing.

The submitted email address and password do not match those on file!

Go back and try again.

图8-34　如果用户没有登录成功，则继续留在登录页

✓提示

- ❑ 如果浏览器已经收到HTTP头，则`headers_sent()`函数会返回TRUE，此时`header()`函数将不能使用。
- ❑ 下面的代码利用GET方法的方式，将值从一个页面传递到另一个页面：

```
$var = urlencode('Pass this text');
header ("Location: page.php? message=$var");
```

- 在重定向时，header()函数从技术上讲应该对目标页使用完整路径。例如，应该是：

```
header ('Location: http://www.example.com/welcome.php');
```

或

```
header ('Location: http://localhost/welcome.php');
```

- 在我的《PHP 6与MySQL 5基础教程》一书中，我给出了一些代码，能够基于当前脚本的位置动态地生成一个绝对URL。

8.10 回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

8.10.1 回顾

- include()与required()的区别是什么？
- 为什么可以将PHP代码放在扩展名是.html的包含文件中？
- 相对引用一个文件与绝对引用一个文件的区别是什么？
- 如何定义一个常量？常量名区分大小写吗？如何检查常量是否已经定义？
- epoch是什么？时间截是什么？
- 以下代码是什么意思：$_SERVER['REQUEST_METHOD']？
- 如何让表单元素“记住”之前提交过的值？
- 如何看到表单元素内出现的PHP错误信息（如显示表单元素的值时）？
- “HTTP头已发送”（headers already sent）错误是什么意思？如何避免？

8.10.2 实践

- 为本章的示例创建一个新的设计原型，之后创建新的header和footer文件。再次浏览站点的任意页面（不需要改变任何PHP脚本）。
- 在header.html中修改date()函数的参数，以不同的方式显示日期和（或）时间。
- 重写register.php中有关密码的条件语句，使用一对嵌套的条件语句。提示：参看第6章的示例。
- 如果使用的是5.2及以上版本的PHP，查找PHP手册中的Filter extension。在register.php中借助filter_var()函数验证Email地址。
- 修改发送注册信息的Email的主题和内容，使其更生动，信息更完整。

第 9 章

cookie和session



本章内容

- ❑ 什么是cookie
- ❑ 创建cookie
- ❑ 读取cookie
- ❑ 向cookie添加参数
- ❑ 删除cookie
- ❑ 创建session
- ❑ 访问session变量
- ❑ 删除session
- ❑ 回顾和实践

第8章讲述了一些较为复杂的Web应用程序开发技术。当开始组建一个多页面的Web站点时很可能遇到的问题之一是HTTP，它是一种无状态技术。这意味着作为一个Web开发者没有内置的方法用以跟踪一个用户或者记录应用程序中的某个页面传递到下一个页面的数据。这是很严重的问题，因为电子商务应用程序、用户注册和登录系统，以及其他常用的在线服务都依赖于这个功能。幸运的是，使用PHP维护从一个页面到另外一个页面的状态是非常容易的。

本章将讨论两种用以跟踪数据的主要方法：cookie和session。从如何创建、读取、修改和删除cookie入手，然后轻松掌握session，更加有效地维护状态。

9.1　什么是 cookie

在cookie出现之前，浏览Web网站是种没有历史可言的“旅程”。虽然浏览器会跟踪所访问的页面，允许使用后退（Back）按钮返回到之前访问过的页面，并且使用不同颜色标注出已经访问过的链接，但是服务器并不会记录谁访问过什么内容。如果站点不使用cookie，或者用户在Web浏览器中禁用了cookie，那么服务器端也不会记录任何内容（参见图9-1）。

为什么这会成为一个问题？如果服务器不能跟踪用户，将不能使用购物车进行在线采购。如果cookie不存在（或者如果它们在Web浏览器中被禁用了），人们将不能使用要求用户注册的那些

流行的站点。简言之，没有cookie就没有Amazon和Facebook或其他流行、有益的站点（或至少不是现在的样子）。



图9-1 大多数Web浏览器让用户能够对处理cookie的参数进行设置。这是Internet Explorer 8高级隐私设置（Advanced Privacy Settings）标签

cookie是服务器在用户计算机上保存用户信息的一种方式，以便服务器能够在访问过程中或者多次访问中记住用户。cookie就像一个名称标签：用户计算机告知服务器用户名称，并且给予一个名称标签。然后服务器能够通过名称标签获知用户是谁。

这引出cookie另外一个有关安全的问题。cookie有不好的一面，这是因为人们认为它让服务器获取了过多关于用户的信息。但是，cookie只存储传递给它的信息，因此它可以达到期望的安全程度。如前所述，现代浏览器都可以根据需要定制cookie处理。

PHP对cookie提供很好的支持。本章将教授如何建立cookie，从cookie中进行信息检索，以及如何删除cookie。还将看到一些可以用来限制cookie的可选参数。

在往下进行之前，需要对cookie了解两点。首先是如何调试与cookie相关的问题。这将在框注中进行讨论。第二点是如何传输和接收cookie（参见图9-2）。cookie存储在Web浏览器中，但是只有最初发送cookie的站点才能够读取它。同时，cookie在Web浏览器对站点中的页面发出请求时被站点读取。换句话说，当用户在地址栏中输入URL地址并点击转到（GO或者类似的其他按钮）时，站点就会读取所有它能访问的cookie并且处理所请求的页面。这个顺序非常重要，这是因为它指明了访问cookie的时间和方式。

✓提示

- 服务器端的PHP和浏览器端的JavaScript都能发送、读取和删除cookie，这是这两种脚本的重叠功能之一。

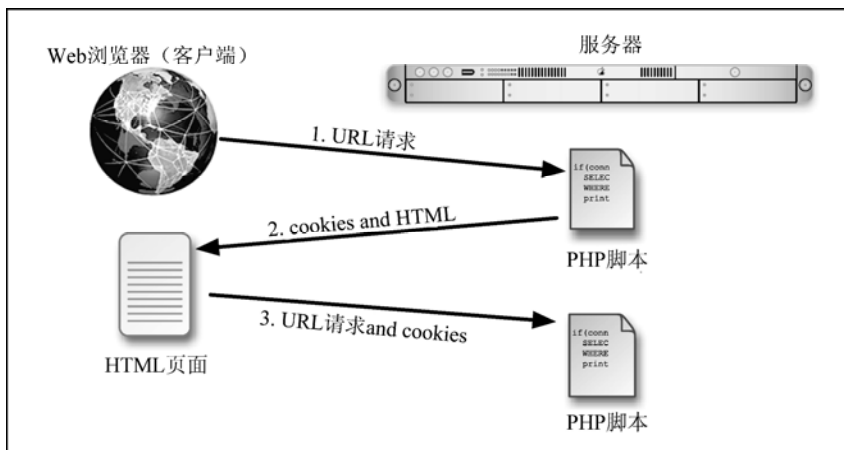


图9-2　在客户端/服务器之间来回传递cookie

调试cookie

在PHP中开始使用cookie时，需要了解当问题发生时如何对脚本进行调试。以下是需要考虑的3个方面：

- 使用PHP发送cookie；
- 在Web浏览器中接受cookie；
- 在PHP脚本中访问cookie。

第一个和最后一个问题能够通过在PHP脚本中打印变量值的方式进行调试（即将学到这些）。第二个问题需要知道如何使用Web浏览器调试cookie。出于调试的目的，将希望Web浏览器在发送cookie时进行通报。

在Windows的Internet Explorer中，可以选择工具菜单中的Internet选项，点击隐私选项卡，选择设置下方的高级按钮。点选“替代自动cookie处理”(参见图9-1)，选择提示第一方和第三方cookie（如果愿意，你也可以拒绝第三方cookie）。Internet Explorer的其他版本对于这个处理的过程大同小异。Internet Explorer还包含一个非常有用的开发者工具（Developer Tools）窗口（位于工具菜单之下）。

使用任何平台的Firefox时，调试cookie的最佳方式是安装cookie相关的扩展包(extensions)，比如Firecookie。Firebug是开发人员必备的扩展包，它也会显示cookie相关的信息。最起码，你应该在Privacy选项卡（在Options或Preferences窗口）中选择Use custom settings for history，这样就可以创建自定义cookie行为，比如前面刚刚提到的行为。

在Mac OS X和Windows平台上的Safari没有提供非常详细的cookie设置，但是在Preferences窗口中的Security选项卡中可以找到设置项。

一些浏览器还能够提供对已经存在的cookie进行浏览的功能，查看它们的名称和值。这些对于调试“战斗”来说是巨大的财富。

9.2 创建 cookie

理解cookie的一个重要事情是，它们必须在发送其他任何信息之前从服务器发送到客户端。也就是说，脚本应该在`print`语句之前，或引入任何包含HTML的外部文件之前发送cookie。

如果服务器尝试在Web浏览器已经获得了HTML代码之后发送cookie，甚至是无关紧要的空格之后，都将产生一个错误消息并且cookie也将发送失败（参见图9-3）。这是迄今为止最常见的与cookie相关的错误。

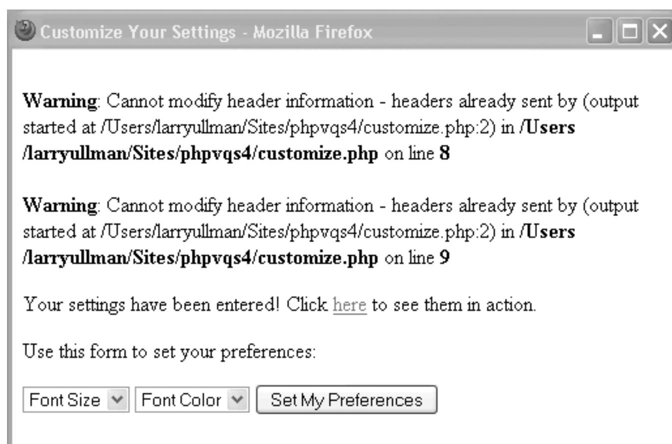


图9-3　如果函数`setcookie()`在任何信息被发送至浏览器之后被调用，甚至发送的是一个空白行，将看到一个类似这样的消息

使用函数`setcookie()`发送cookie：

```
setcookie(name, value);
setcookie('CookieName', 'This is the cookie value.');
```

这行代码向浏览器发送了一个名为`CookieName`的cookie，其值为`This is the cookie value`（参见图9-4）。

尽管向相同站点发送cookie的数量已经被Web浏览器所限制，不过还是可以用函数`setcookie()`继续向浏览器发送更多的cookie。

```
setcookie('name2', 'some value');
setcookie('name3', 'another value');
```

最后，正如即将在本示例中看到的那样，当创建cookie时，可以使用一个变量作为cookie的名称或者值属性。

```
setcookie($cookie_name, $cookie_value);
```

作为设置cookie的示例，这里将创建一个脚本以让用户来指定页面文本的字体大小和颜色。页面将显示在表单中选择这些值的结果（参见图9-5），然后提交表单（参见图9-6）。本章的9.3节将创建出一个单独的页面，它将用到这些设置。

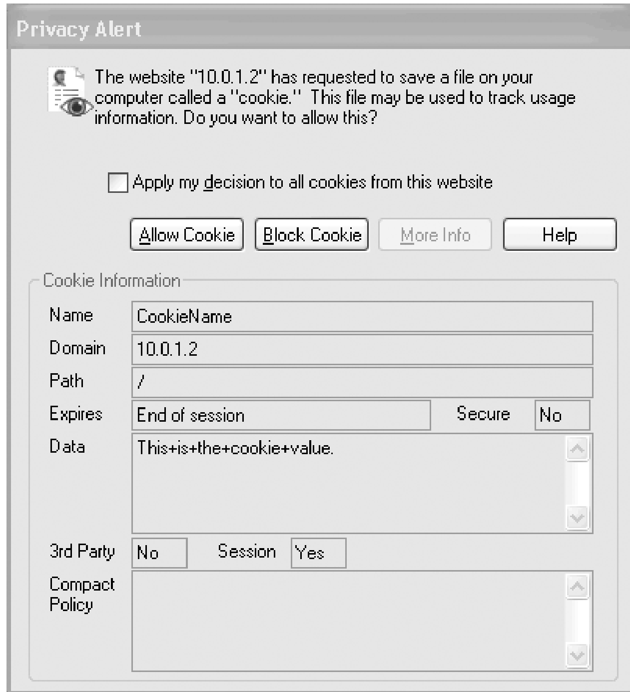


图9-4　如果浏览器被设置为提示用户使用了cookie，在每次发送cookie时将会出现一个这样的消息（注意Internet Explorer 8的这个窗口显示了URL编码格式中的值）

Use this form to set your preferences:

Font Size　Font Color　Set My Preferences

图9-5　该表单用于选择另外一个PHP页面使用的字体大小和颜色

Your settings have been entered! Click here to see them in action.

Use this form to set your preferences:

Font Size　Font Color　Set My Preferences

图9-6　在提交表单之后，页面显示出一条消息和指向另外一个页面（将使用用户选择的页面）的链接。接下来将创建这个页面

⇒ 发送cookie

(1) 在文本编辑器或者IDE中创建一个新的PHP文档，命名为customize.php（参看脚本9-1）：

```
<?php // 脚本9-1 - customize.php
```

脚本9-1 两个cookie将用来存储用户对于文本的字体大小和颜色的选择。该页面将显示并处理这个表单

```
1   <?php // 脚本9-1 - customize.php
2
3   // 如果表单已提交，则处理它：
4   if (isset($_POST['font_size'], $_POST['font_color'])) {
5
6     // 发送cookie：
7     setcookie('font_size', $_POST['font_size']);
8     setcookie('font_color', $_POST['font_color']);
9
10    // 打印一条消息：
11    $msg = '<p>Your settings have been entered! Click <a href="view_settings.php">
      here</a> to see them in action.</p>';
12
13  } // 结束条件语句。
14  ?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
15    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
16  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
17  <head>
18    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
19    <title>Customize Your Settings </title>
20  </head>
21  <body>
22  <?php // 如果cookie已发送，打印一条消息。
23  if (isset($msg)) {
24    print $msg;
25  }
26  ?>
27
28  <p>Use this form to set your preferences:</p>
29
30  <form action="customize.php" method="post">
31    <select name="font_size">
32    <option value="">Font Size</option>
33    <option value="xx-small">xx-small </option>
34    <option value="x-small">x-small </option>
35    <option value="small">small </option>
36    <option value="medium">medium </option>
37    <option value="large">large </option>
38    <option value="x-large">x-large </option>
39    <option value="xx-large">xx-large </option>
40    </select>
41    <select name="font_color">
42    <option value="">Font Color </option>
43    <option value="999">Gray</option>
44    <option value="0c0">Green</option>
45    <option value="00f">Blue</option>
46    <option value="c00">Red</option>
47    <option value="000">Black</option>
48    </select>
49    <input type="submit" name="submit" value="Set My Preferences" />
```

```
50  </form>
51
52  </body>
53  </html>
```

cookie最重要的问题是它们在向Web浏览器发送任何信息之前被创建。为了实现这个目的，这些脚本位于处理cookie发送的PHP代码之前。

同时请确认PHP起始标签之前不要有空格或空行。

(2) 检验表单是否被提交：

```
if (isset($_POST['font_size'], $_POST['font_color'])) {
```

该页面会显示并处理表单。它会使用第8章介绍的方法（即检查`$_SERVER['REQUEST_METHOD']`变量的值是否为POST）验证表单是否提交，但这里采用的是另外一种方式，实现最基本、最小限度的验证。条件语句检查是否已经设置了`$_POST['font_size']`和`$_POST['font_color']`变量。如果两个变量都已设置，就表明表单已提交。

(3) 创建cookie：

```
setcookie('font_size', $_POST['font_size']);
setcookie('font_color', $_POST['font_color']);
```

这两行代码创建了两个单独的cookie：一个名为`font_size`，另一个名为`font_color`。它们的值将取决于HTML表单中所选的值，这些值被存贮在变量`$_POST['font_size']`和`$_POST['font_ color']`中。

在实际开发的应用程序中，我会在cookie使用表单提交的元素之前，对用户选择的值进行验证。

(4) 创建一条信息，并且完成条件语句和PHP代码段：

```
  $msg = '<p>Your settings have been entered! Click <a href=
  "view_settings.php">here</a> to see them in action.</p>';
} // 结束条件语句。
?>
```

当表单被提交时，将发送cookie，并且变量`$msg`会被赋予一个字符串类型的值。这个变量稍后在脚本中会被使用并用以打印一条消息。这一步很重要，因为不能在连接处（这时还没有创建HTML头）打印信息。

(5) 创建HTML头并且开启body标签：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Customize Your Settings </title>
</head>
<body>
```

所有的这些代码都必须在`setcookie()`行之后。这里并没有夸大事实，没有文本、HTML或者空白能够在调用`setcookie()`之前发送至Web浏览器。

(6) 创建另外一个PHP代码段以报告cookie已经被发送：

```
<?php
if (isset($msg)) {
  print $msg;
}
?>
```

如果cookie已经被发送，这些代码将会打印出一条消息。用户第一次访问页面时，cookie并不会被发送，因此不会设置`$msg`，这个条件的结果为FALSE，并且`print`调用不会运行。一旦提交表单，就会设置`$msg`，这个条件的结果将为TRUE。

(7) 开始编写HTML表单：

```
<p>Use this form to set your preferences:</p>
<form action="customize.php" method="post">
<select name="font_size">
<option value="">Font Size</option>
<option value="xx-small">xx-small </option>
<option value="x-small">x-small </option>
<option value="small">small</option>
<option value="medium">medium </option>
<option value="large">large</option>
<option value="x-large">x-large </option>
<option value="xx-large">xx-large </option>
</select>
```

HTML表单非常简单（参见图9-5）。用户可以通过一个下拉菜单来选择字体大小。每个值都对应CSS代码以用来设置文档的字体大小：从`xx-small`到`xx-large`。

因为这段脚本显示并处理表单，所以表单的行为属性将指向相同的这个文件。

(8) 完成HTML表单：

```
  <select name="font_color">
  <option value="">Font Color </option>
  <option value="999">Gray</option>
  <option value="0c0">Green</option>
  <option value="00f">Blue</option>
  <option value="c00">Red</option>
  <option value="000">Black</option>
  </select>
  <input type="submit" name= "submit" value="Set My Preferences" />
</form>
```

第二个下拉菜单用来选择字体颜色。菜单显示文本表单中的颜色，但是其值为HTML类型的颜色值。通常这样的值使用六个字符外加一个#号来表示（如`#00CC00`），但是CSS允许仅使用3个字符的版本，并且将在使用这些值的页面中添加#号。

(9) 完成HTML页：

```
</body>
</html>
```

(10) 将文件保存为`customize.php`，并且放置在启用了PHP服务器上适当的目录中。

9

(11) 确保已经将Web浏览器设置为为每个cookie进行提示。

为了保证脚本能够运行，设置浏览器为每个cookie提供提示。参见“调试cookie”框注。

(12) 在Web浏览器中运行脚本（参见图9-7和图9-8）。

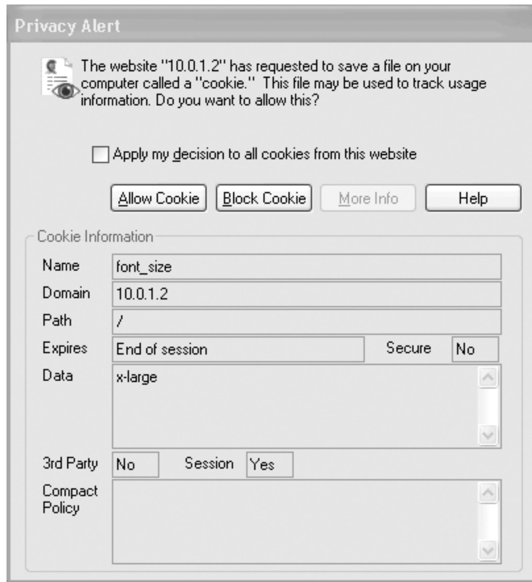


图9-7　如果用户已经选择了在接受cookie前进行提示，那么当`setcookie()`第一次被调用时，他们将看到这条消息。存储名为`font_size`且值为`x-small`的cookie

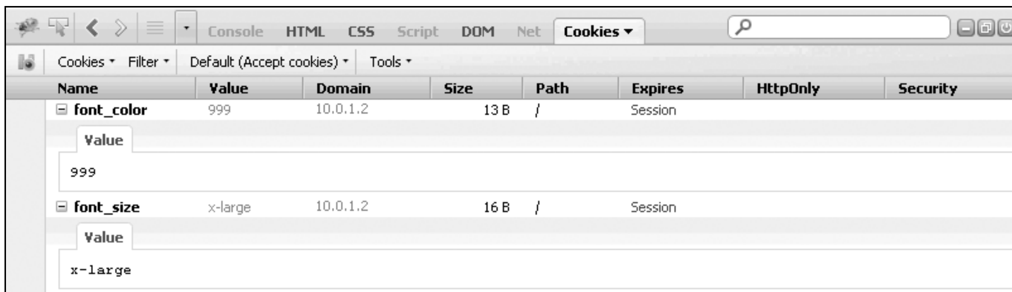


图9-8　Firefox的Firebug扩展包显示了浏览器接收到的cookie。第2个cookie是由PHP脚本发送的，名为`font-color`值为`999`，代表字体颜色为灰色

✓提示

- ❑ cookie是PHP中为数不多的几个因为浏览器的不同或操作系统的不同而不同的工具之一。应当在尽可能多的浏览器和操作系统中测试基于cookie的应用程序。
- ❑ 如果使用第8章中教授的输出缓存技术，那么就可以将`setcookie()`调用放置在脚本中的任何位置（这是因为Web浏览器在调用函数`ob_end_flush()`之前不会获得数据）。

- cookie的总数限制接近于4KB。这已经远远超出大多数应用程序所需的数量。
- 可以使用`headers_sent()`函数来测试发送cookie是否安全。它将报告HTTP头是否已经发送给Web浏览器。

9.3　读取 cookie

就像表单数据被存储在数组`$_POST`中（假定该表单使用POST方法），URL中传送给脚本的值被存储在数组`$_GET`中那样，函数`setcookie()`将cookie数据存放在数组`$_COOKIE`中。从cookie中获得某个值，只需要将cookie的名称指定为该数组的索引即可。例如，获得在下面脚本创建的cookie值：

```
setcookie('user', 'trout');
```

可以使用变量`$_COOKIE['user']`。

除非更改了cookie的参数（将在本章稍后部分看到），Web应用程序中每个页面都可以访问cookie。但是应该知道，cookie在未被发送前不能立即被脚本访问。这里不能这样做：

```
setcookie('user', 'trout');
print $_COOKIE['user']; // No value.
```

其中的原因是cookie被读取和发送的顺序问题（参见图9-2）。

访问cookie的值非常简单，我们编写一段脚本来使用在`customize.php`中用来指定页面文本大小和颜色的偏好设置。该脚本将使用CSS以达到这样的效果。

⇛ 用PHP检索cookie数据

(1) 在文本编辑器或者IDE中开启一个新的PHP文档，命名为`view_settings.php`（参看脚本9-2）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>View Your Settings</title>
```

脚本9-2　该脚本使用在cookie中存储的值设置CSS中的字体大小和颜色

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>View Your Settings</title>
7     <style type="text/css">
8       body {
9   <?php // 脚本9-2 - view_settings.php
10
```

9

```
11 // 检查font_size的值:
12 if (isset($_COOKIE['font_size'])) {
13   print "\t\tfont-size: " . htmlentities($_COOKIE['font_size']) . ";\n";
14 } else {
15   print "\t\tfont-size: medium;";
16 }
17
18 // 检查font_color的值:
19 if (isset($_COOKIE['font_color'])) {
20   print "\t\tcolor: #" . htmlentities($_COOKIE['font_color']) . ";\n";
21 } else {
22   print "\t\tcolor: #000;";
23 }
24
25 ?>
26       }
27   </style>
28 </head>
29 <body>
30 <p><a href="customize.php">Customize Your Settings</a></p>
31 <p><a href="reset.php">Reset Your Settings</a></p>
32
33 <p>yadda yadda yadda yadda yadda
34 yadda yadda yadda yadda yadda
35 yadda yadda yadda yadda yadda
36 yadda yadda yadda yadda yadda
37 yadda yadda yadda yadda yadda</p>
38
39 </body>
40 </html>
```

(2) 开始CSS代码段：

```
<style type="text/css">
  body {
```

这个页面将使用CSS来实现用户的偏好设置。那么要完成的脚本将类似：

```
body {
  font-size: x-large;
  color: #999;
}
```

字体大小和颜色的值将根据用户在customize.php页面中的选择而不同。在这一步中，创建了初始的CSS代码。

(3) 开启PHP代码段：

```
<?php // 脚本9-2 - view_settings.php
```

该脚本将基于cookie用PHP来打印剩余的CSS。

(4) 如果cookie存在的话，使用cookie：

```
if (isset($_COOKIE['font_size'])) {
  print "\t\tfont-size: " . htmlentities($_COOKIE['font_size']) . ";\n";
} else {
```

```
  print "\t\tfont-size: medium;";
}
```

如果脚本能够访问包含font_size设定的cookie，它将打印出该cookie的值以作为CSS字体大小的值。使用isset()函数是很必要的，可以用来查看cookie是否存在。如果不存在这样的cookie，PHP将以默认大小（medium）打印。

出于安全考虑，cookie的值不会被直接打印出来。取而代之的做法是，运行通过在第5章中所讨论的函数htmlentities()来阻止因为用户对cookie的值进行操纵（这样做非常容易）而可能造成的不良后果。

注意，每个print语句都将创建两个制表符（\t）和一个换行符（\n）。这样做的目的是让将在HTML源代码中出现的CSS代码以某种格式显示出来。如果不这样做也不会影响页面的功能，只是……

(5) 为保存字体颜色的cookie重复这个过程：

```
if (isset($_COOKIE['font_color'])) {
  print "\t\tcolor: #" . htmlentities($_COOKIE['font_color']) . ";\n";
} else {
  print "\t\tcolor: #000; ";
}
```

这里的CSS颜色属性已经被赋予了一个值。在第4步中cookie本身也被相同的方式使用。

(6) 结束PHP代码段，完成CSS代码并结束HTML的head部分：

```
?>
    }
  </style>
</head>
```

(7) 开始编写HTML的body部分，并为两个页面创建链接：

```
<body>
<p><a href="customize.php">Customize Your Settings</a></p>
<p><a href="reset.php">Reset Your Settings</a></p>
```


这两个链接将用户引向另外的两个PHP页面。第一个是已经编写出来的customize.php，它用来让用户自定义设置。第二个页面是reset.php，本章稍后将编写它，这个页面允许用户对他们自定义的设置进行删除。

(8) 添加一些文本：

```
<p>yadda yadda yadda yadda yadda
yadda yadda yadda yadda yadda
yadda yadda yadda yadda yadda
yadda yadda yadda yadda yadda
yadda yadda yadda yadda yadda</p>
```

该文本用来显示cookie变更带来的效果。

(9) 完成HTML页面：

```
</body>
</html>
```

(10) 将文件保存为`view_settings.php`，放置在同`customize.php`相同的目录中，然后单击`customize.php`上的链接，在Web浏览器中测试（参见图9-9）。



图9-9　该页面反应了用PHP脚本所作的自定义字体的选择

(11) 查阅页面的源代码以查看最终的CSS代码（参见图9-10）。



图9-10　通过查阅页面的源代码，还可以对CSS值的变更进行跟踪

(12) 使用自定义页面来更改设置并且返回给这段脚本。

每次提交表单都将创建两个用来存储表单值的新的cookie，从而替换已经存在的cookie。

✓提示

- cookie的值在它被发送时将被自动编码，并且在被PHP页面获得的时候被自动解码。对于HTML表单发送的值也是如此。

9.4 向 cookie 添加参数

虽然向函数setcookie()只传送name和value参数对于大多数情况下的cookie使用来说已经足够，但是还应当知道还有其他的可用参数。函数接受的参数能够多达5个，每个参数都用于限制对cookie的操作：

```
setcookie(name, value, expiration, path, domain, secure, httponly);
```

参数expiration用来为cookie的存在设定一个特定的时间长度。如果它没有被指定，cookie将一直起作用直到用户关闭他们的浏览器。通常情况下，通过向当前时间加上一个特定数量的分钟或者小时数来指定过期时间。使用PHP的time()函数（参见第8章）获取当前时间。下面这行代码将cookie的过期时间设定为当前时间之后的1小时（60秒乘以60分钟）：

```
setcookie(name, value, time()+3600);
```

因为过期时间将被计算为time()的值加上3600后的结果，这个特别的参数没有被引号引用（因为不希望逐字传送time()+3600作为过期时间，而是传送计算结果）。

path和domain参数用来限制在Web站点（路径）中的特定文件夹或者特定域中的cookie。使用path选项，可以限制仅当用户在域中user文件夹中时cookie才存在：

```
setcookie(name, value, time()+3600, '/subfolder/');
```

cookie已经被指定给一个域，因此域参数可以用来限制一个子域的cookie，如forum.example.com。

```
setcookie(name, value, time()+3600, '', 'forum.example.com');
```

参数secure的值指明一个cookie应当只能通过安全HTTPS连接传送。值1指明必须使用安全连接，反之值0指明安全连接并不必要。可以为一个电子商务站点确认一个安全cookie传输：

```
setcookie('cart', '82ABC3012', time()+3600, ', 'shop.example.com', 1);
```

和所有带有参数的函数一样，必须按顺序传递所有的参数值。在先前的示例中，如果不希望指定（或者限制）路径和域，需要使用空的引号。在path参数中，可以使用一个单独的斜线（/）表示根目录（即，无路径重定向）。通过这样做，可以保留适当数量的参数并且如果需要仍然能够指定使用HTTPS连接。

最后一个属性（httponly）是PHP5.2版本中新添加的属性。它可以用来限制对cookie的访问（例如，防止用JavaScript读取cookie），但不是所有的浏览器都对此提供支持。

让我们为已经存在的customize.php页面添加过期日期，以便用户在关闭了浏览器之后以及重新返回之前访问的站点时，仍然保留他们的首选项。

⇛ 为cookie设置过期日期

(1) 在文本编辑器或者IDE中打开customize.php。(参看脚本9-1)。

(2) 按照下面的代码修改2行setcookie()（参看脚本9-3）：

```
setcookie('font_size', $_POST['font_size'], time()+10000000, '/', '', 0);
setcookie('font_color', $_POST['font_color'], time()+10000000, '/', '', 0);
```

脚本9-3　通过向这两个cookie添加expiration参数的方式，甚至可以让cookie持续到用户关闭和重新返回到浏览器之后

```
1   <?php // 脚本9-3 - customize.php #2
2
3   // 如果表单已提交，则处理它：
4   if (isset($_POST['font_size'], $_POST['font_color'])) {
5
6     // 发送cookie：
7     setcookie('font_size', $_POST['font_size'], time()+10000000, '/');
8     setcookie('font_color', $_POST['font_color'], time()+10000000, '/');
9
10    // 打印一条消息：
11    $msg = '<p>Your settings have been entered! Click <a href="view_settings.php">
      here</a> to see them in action.</p>';
12
13  } // 结束条件语句。
14  ?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
15    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
16  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
17  <head>
18    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
19    <title>Customize Your Settings </title>
20  </head>
21  <body>
22  <?php // 如果cookie已发送，打印一条消息。
23  if (isset($msg)) {
24    print $msg;
25  }
26  ?>
27
28  <p>Use this form to set your preferences:</p>
29
30  <form action="customize.php" method="post">
31    <select name="font_size">
32    <option value="">Font Size</option>
33    <option value="xx-small">xx-small </option>
34    <option value="x-small">x-small </option>
35    <option value="small">small </option>
36    <option value="medium">medium </option>
37    <option value="large">large </option>
38    <option value="x-large">x-large </option>
39    <option value="xx-large">xx-large </option>
```

```
40     </select>
41     <select name="font_color">
42     <option value="">Font Color</option>
43     <option value="999">Gray</option>
44     <option value="0c0">Green</option>
45     <option value="00f">Blue</option>
46     <option value="c00">Red</option>
47     <option value="000">Black</option>
48     </select>
49     <input type="submit" name="submit" value="Set My Preferences" />
50   </form>
51
52   </body>
53   </html>
```

为了让cookie持续一个较长的时间（持续数个月），可以设定过期时间为从当前时间开始往后10 000 000秒。此时，设置`path`参数指向网站的根目录（使用`/`）。这样做将可以改善通过不同浏览器进行cookie发送时的一致性。

因为cookie的过期时间被设置为将来的数月之后，保存在cookie中的用户偏好设置在用户关闭以及重新打开浏览器时仍然有效。如果没有过期时间，用户将看到默认的字体大小和颜色，并且必须为每个浏览器会话重新分配偏好设置。

(3) 保存文件，放置在启用了PHP的服务器上适当的目录中，并在Web浏览器中重新测试（参见图9-11和图9-12）。



图9-11　如果将浏览器设置为当接受到cookie时提供提示，将看到过期时间是如何被添加的（请同图9-6进行对比）

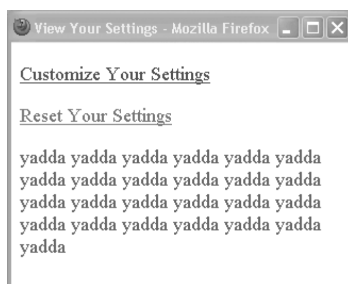


图9-12　新的cookie参数并没有对应用程序的功能起到相反的作用

✓提示

- 请注意，当从自己的计算机（如从localhost）发送cookie时，有的浏览器也许不能对cookie调整后的过期时间作出反应。
- 对于在使用cookie时选用何种过期时间来说有几个常用的原则。如果cookie持续的时间同用户浏览网站的时间一样长，那么不要设置过期时间。如果cookie需要在用户关闭和重新开启浏览器之后继续存在，那么将过期时间设置为数月后的将来。如果cookie会造成一些安全隐患，将过期时间设置为一个小时，或者一小段时间，这样cookie不会在用户离开浏览器之后很久依旧存在。
- 当服务器和客户端在不同的时区内时，cookie的过期时间将会造成较大的麻烦。过期时间是基于服务器的时区而设定的，但是浏览器可能会删除基于客户端时区的cookie。幸运的是，一些浏览器能够自动对此进行正确操作。
- 出于安全考虑，可以为cookie设定一个5分钟或10分钟的过期时间，并且让cookie在用户每访问一个新页面时就被重新发送一次。通过这种方式，cookie能够在用户访问的全程存在，而在用户离开后5分钟或10分钟后消失。

9.5　删除 cookie

使用cookie需要了解的最后一件事情是如何删除它。虽然cookie会在用户的浏览器关闭或者过期日期/时间到达时自动过期，但是通常还是需要能够手动删除它。例如，用来进行用户注册和登录的Web站点通常都能够在用户注销时删除所有的cookie。

函数setcookie()接受7个参数，但是只有name这个参数是必须的。如果传送一个有名称而无值的cookie，它起的作用同删除一个已经存在的同名cookie一样。例如，使用下面的代码来创建名为username的cookie：

```
setcookie('username', 'Larry');
```

删除名为username的cookie，可以编写代码

```
setcookie('username', '');
```

或

```
setcookie('username', FALSE);
```

为了更加谨慎，还可以为之设定一个在过去的过期时间（参见图9-13）：

```
setcookie('username', FALSE, time() - 600);
```

当删除cookie时唯一需要告诫的是，必须使用同首次设置cookie时相同的参数值（除了值和过期时间）。例如，如果在创建cookie时提供了domain值，那么在删除该cookie的时候也应该提供同样的值：

```
setcookie('user', 'larry', time() +  3600, '', 'forums.example.com');
setcookie('user', '', time() - 600, '', 'forums.example.com');
```

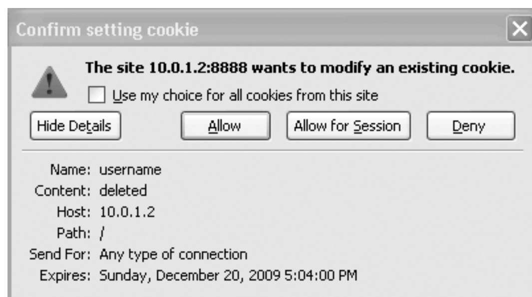


图9-13　当发送删除cookie时，Firefox显示的cookie信息

诠释这个特性，我们为Web应用程序添加一个重置页面，它将摧毁已发送的cookie以删除用户的偏好设置。

⇛ 删除一个cookie

(1) 在文本编辑器或者IDE中开启一个新的PHP脚本，命名为reset.php（参看脚本9-4）：

```
<?php // 脚本9-4 - reset.php
```

脚本9-4　发送同已有cookie同名的空白cookie和过去的过期时间，重设所有的cookie的值

```
1   <?php // 脚本9-4 - reset.php
2
3   // 删除cookie：
4   setcookie('font_size', '', time()-  600, '/');
5   setcookie('font_color', '', time()- 600, '/');
6
7   ?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
8     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
9   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
10  <head>
11    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
12    <title>Reset Your Settings</title>
13  </head>
14  <body>
15
16  <p>Your settings have been reset! Click <a href="view_settings.php">here</a> to
    go back to the main page.</p>
17
18  </body>
19  </html>
```

(2) 使用发送空白cookie的方式删除已有的cookie，并完成PHP代码：

```
setcookie('font_size', '', time() - 600, '/');
setcookie('font_color', '', time() - 600, '/');
?>
```

这两行代码发送了名为font_size和font_color的两个cookie，两个都没有值，并且过期时间都在10分钟以前。正如在创建cookie时那样，必须在任何其他信息被发送至Web浏览器之前

调用函数setcookie()。

(3) 创建HTML的head部分：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Reset Your Settings</title>
</head>
```

(4) 添加页面的body部分：

```
<body>
<p>Your settings have been reset! Click <a href="view_settings.php">
→here</a> to go back to the main page.</p>
</body>
```

脚本中的body部分仅仅用来告诉用户他们的设置已经被重置。现在提供了一个链接来引导他们返回到主页。

(5) 完成HTML：

```
</html>
```

(6) 将页面保存为reset.php，放置在启用了PHP的服务器上适当的目录中，并在浏览器中测试（参见图9-14、图9-15和图9-16）。

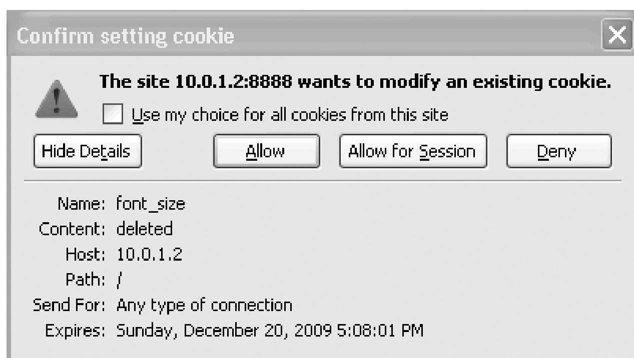


图9-14　当对只有名称没有值的cookie使用函数setcookie()时，已经存在的同名cookie将被删除。设置为过去的过期时间也将会确保正确删除已存在的cookie



图9-15　重置页面发送两个空cookie并将显示这样的消息

图9-16　如果用户使用了这个重置页面，PHP将摧毁这个cookie（参见图9-14），并将重置主页为它默认的格式

单击`view_settings.php`（参看脚本9-2）中适当的链接，测试该页面，或者直接前往该页。

✓提示

- 在加载另外的页面之后cookie才会生效。类似地，删除一个cookie也需要等到加载另外的页面之后。这就是说可以删除一个页面中的cookie，但是仍然可以对之进行访问（这是因为页面获取cookie是在删除cookie之前）。
- 如同使用不同的浏览器将会呈现创建cookie的不同结果那样，删除cookie也会造成同样的状况。请在多个浏览器中测试脚本，并且使用`setcookie()`设置以确保最佳最全面的兼容性。

9.6　什么是 session

session就像是cookie，它是一种解决方案，可以跟踪用户在一系列页面中访问的数据。两者之间非常重要的区别在于，cookie将数据保存在客户端（在Web浏览器中），而session则将数据保存在服务器端。因为有这个区别，session拥有比cookie更多的优势。

- session通常情况下更加安全，这是因为数据不会在客户端和服务器端来回重复传输。
- session能够存储比cookie更多的信息。
- session甚至可以在用户不接受他们浏览器中的cookie时仍然能够工作。

当开启一个session时，PHP将会创建一个随机的session ID。每个用户的session都会有一个自己的session ID，与服务器上存储该用户session数据的文本文件名相同（参看脚本9-5）。

这样网站上所有PHP脚本都可能使用某一用户的session数据，同时还会追踪session ID。默认情况下，这个session ID将作为一个cookie发送给Web浏览器（参见图9-17）。接下来PHP页面将使用这个cookie来检索session ID并访问session信息。

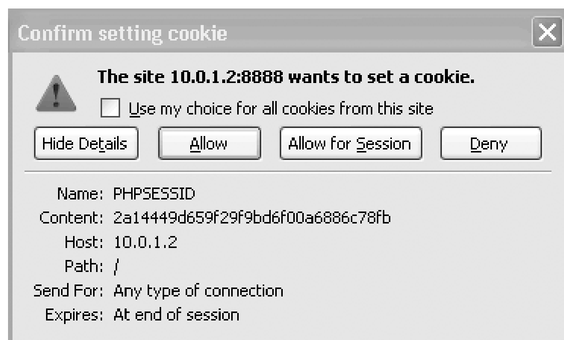


图9-17　一个带有session信息的cookie被发送给Web浏览器

在下面几页中，将看到使用PHP可以非常容易地操作session。

在session和cookie中进行选择

session同cookie相比有很多优势，但是在一些情况下仍然需要选择后者。cookie在以下方面相比session更有优势：

- 更加易于创建和检索；
- 对服务器造成的处理压力更少；
- 通常情况下能够持续更长的时间。

简而言之，在安全问题不是那么突出，并且只有最少量的数据需要存储时需要使用cookie。如果需要考虑安全问题，并且有许多信息需要保存，最好使用session。但是使用session会在编写脚本的时候增加一点工作量。

9.7　创建 session

可以使用函数`session_start()`创建、访问或删除session。这个函数将试图在session首次启动时发送一个cookie，因此它必须在任何HTML或空白被发送至Web浏览器之前调用。因此，在使用session的页面中，必须在脚本的起始行调用函数`session_start()`：

```
<?php
session_start();
```

当第一次开启session时，会产生一个随机的session ID，并且会向Web浏览器发送一个名为PHPSESSID（session的名称）的cookie，它的值看上去类似于4bcc48dc87cb4b54d63f99da23fb41e1（参见图9-18）。

一旦启用session，可以通过向数组`$_SESSION`赋值的方式记录数据：

```
$_SESSION['first_name'] = 'Sam';
$_SESSION['age'] = 4;
```

这个数组与PHP中使用的其他数组不同，它必须是关联数组。换句话说，应该显式地使用字

符串作为键，如first_name和age。

每次这样做时，PHP将向服务器上的一个临时文件中编写一些数据（参看脚本9-5）。

脚本9-5　session数据在服务器文件中的存储方式

```
1   email|s:14:"me@example.com"; loggedin|i:1292883103;
```

首先我们将重写第8章中的登录脚本，这次要将Email地址保存在一个session中。

⇛ 创建一个session

(1) 在文本编辑器或者IDE中打开login.php（参看脚本8-13）。

(2) 在ob_end_clean()代码行之前，添加下面的代码（参看脚本9-6）：

```
session_start();
$_SESSION['email'] = $_POST['email'];
$_SESSION['loggedin'] = time();
```

脚本9-6　该脚本保存了session中的两个值，并且将用户重新定向至其session值被访问的其他页面

```
1   <?php // 脚本9-6 - login.php #3
2   /* 页面可以用于用户登录（基本完成!）。*/
3
4   // 设置页面标题并包含页头文件:
5   define('TITLE', 'Login');
6   include('templates/header.html');
7
8   // 打印一些介绍文本:
9   print '<h2>Login Form</h2>
10    <p>Users who are logged in can take advantage of certain features like this, that,
      and the other thing.</p>';
11
12  // 检查表单是否已提交:
13  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
14
15    // 处理表单:
16    if ( (!empty($_POST['email'])) && (!empty($_POST['password'])) ) {
17
18      if ( (strtolower($_POST ['email']) == 'me@example.com') && ($_POST['password'] ==
        'testpass') ) { // 相等!
19
20        // 开启session缓冲:
21        session_start();
22        $_SESSION['email'] = $_POST['email'];
23        $_SESSION['loggedin'] = time();
24
25        // 将用户重定向到欢迎页!
26        ob_end_clean(); // 消除缓冲!
27        header ('Location: welcome.php');
28        exit();
29
30      } else { // 不相等!
31
32        print '<p>The submitted email address and password do not match those on
          file!<br/>Go back and try again.</p>';
33
34      }
```

```
35
36     } else { // 表单未填写完整。
37
38       print '<p>Please make sure you enter both an email address and a password!<br
         />Go back and try again.</p>';
39
40     }
41
42 } else { // 显示表单。
43
44    print '<form action="login.php" method="post">
45    <p>Email Address: <input type="text" name="email" size="20" /></p>
46    <p>Password: <input type="password" name="password" size="20" /></p>
47    <p><input type="submit" name= "submit" value="Log In!" /></p>
48    </form>';
49
50 }
51
52 include('templates/footer.html'); // 包含页脚文件。
53 ?>
```

可以通过调用函数`session_start()`以在session中保存值。虽然通常必须在脚本中首先调用此函数（因为它将试图发送一个cookie），但是在这里并不是必须的，这是因为该脚本的`header`文件开启了输出缓存（参见第8章）。

session将首先在`$_SESSION['email']`中保存用户提交的Email地址。然后将用户登录的时间赋值给`$_SESSION['loggedin']`。这是由调用函数`time()`所决定的，它将返回从epoch（1970年1月1日午夜）开始过去的秒数。

(3) 将文件保存为`login.php`，放置在启用了PHP的计算机上适当的目录中。

该脚本应该被放置在同第8章中所述相同的目录中，因为它需要其中一些其他的文件。

(4) 在Web浏览器中加载该表单以确保它没有错误发生（参见图9-18）。

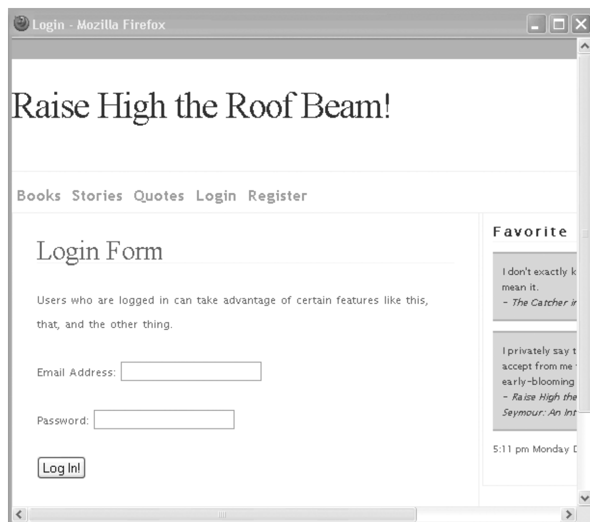


图9-18　登录表单

没有完成和提交登录表单，因为欢迎页面需要在实际登录之前更新。

✓提示

- php.ini配置文件包含许多同session相关的设置，如果具有服务器管理员级别的权限，就可以修改php.ini文件。在文本编辑器中打开php.ini文件以查看提供更多信息的手册。
- 还可以使用ini_set()函数来更改一些session设置。
- session_name()函数可以修改session的名称（替代默认的PHPSESSID）。它必须在每次调用session_start()之前使用，就像这样：

```
session_name('YourVisit');
session_start();
```

- session_set_cookie_params()函数用来修改session cookie的设置（如过期时间、路径和域）。
- 常量SID用格式name=ID的方式保存一个字符串。例如：

```
PHPSESSID=4bcc48dc87cb4b54d63f99da23fb41e1
```

- 可以在session中保存任何类型的值，如数值、字符串、数组或者对象，甚至是它们的任意组合。

9.8 访问 session 变量

我们已经在session中保存了值，接下来需要学习如何访问这些值。不论是创建了新的session或者访问一个已经存在的session，都必须从函数session_start()开始。该函数向PHP指明这个特殊的脚本将要使用session。

引用$_SESSION变量非常简单，和引用其他数组的操作差不多。考虑到这一点，这里将编写另外一个同第8章类似的欢迎页面，用来访问存储的Email和loggedin的值。

⇒ 访问session变量的步骤

(1) 在文本编辑器或者IDE中创建一个新的PHP文档，命名为welcome.php（参看脚本9-7）：

```
<?php // 脚本9-7 - welcome.php
```

脚本9-7 只要脚本首先使用session_start()，就可以通过使用数组$_SESSION来访问已保存的session值

```
1   <?php // 脚本9-7 - welcome.php #2
2   /* 这是欢迎页，用户成功登录后
3   会被重定向到这里。*/
4   
5   // 开始session部分：
6   session_start();
7   
8   // 设置页面标题并包含头文件：
9   define('TITLE', 'Welcome to the J.D. Salinger Fan Club!');
```

9

```
10  include('templates/header.html');
11
12  // 打印欢迎信息：
13  print '<h2>Welcome to the J.D. Salinger Fan Club!</h2>';
14  print '<p>Hello, ' . $_SESSION ['email'] . '!</p>';
15
16  // 打印用户登录时间：
17  date_default_timezone_set('America/New_York');
18  print '<p>You have been logged in since: ' . date('g:i a', $_SESSION['loggedin']) .
    '</p>';
19
20  // 创建登出链接：
21  print '<p><a href="logout.php">Click here to logout.</a></p>';
22
23  include('templates/footer.html'); // 包含页脚文件。
24  ?>
```

(2) 开始编写session部分：

```
session_start();
```

当访问session值的时候，应该在将任何数据发送给Web浏览器之前调用函数session_start()。

(3) 定义页面的标题，并且包含HTML的header部分：

```
define('TITLE', 'Welcome to the J.D. Salinger Fan Club!');
include('templates/header.html');
```

由于该页使用的模板系统同第8章中所开发的相同，因此它也将使用同样的header系统。

(4) 使用用户的Email地址显示问候信息：

```
print '<h2>Welcome to the J.D. Salinger Fan Club!</h2>';
print '<p>Hello, ' . $_SESSION ['email'] . '!</p>';
```

可以通过引用$_SESSION['email']来访问已保存的用户地址。在这里，该值已同正在打印的字符串剩余部分连接。

(5) 显示用户已经登录的时长：

```
date_default_timezone_set ('America/New_York');
print '<p>You have been logged in since: ' . date('g:i a', $_SESSION['loggedin']) . '</p>';
```

可以通过引用$_SESSION['loggedin']变量的方式显示用户已经登录的时长。通过将之作为发送给date()函数的第二个参数，连同具有适当格式的参数，可以使用PHP脚本创建类似于11:22pm这样的文本。

但是在使用date()函数之前，需要设定默认时区（这点在第8章中也讨论过）。如果愿意，在设置了时区之后，可以从页脚文件中移除对该函数的调用。

(6) 完成内容部分：

```
print '<p><a href="logout.php">Click here to logout.</a></p>';
```

下一段脚本将提供注销功能，因此这里需要添加一个链接。

(7) 将HTML页脚包含进来，并且完成HTML页面：

```
require('templates/footer.html');
?>
```

(8) 将文件保存为welcome.php，放置在启用了PHP服务器上适当的目录中，并且在Web浏览器中（参见图9-19）测试（从脚本9-6中login.php开始）。

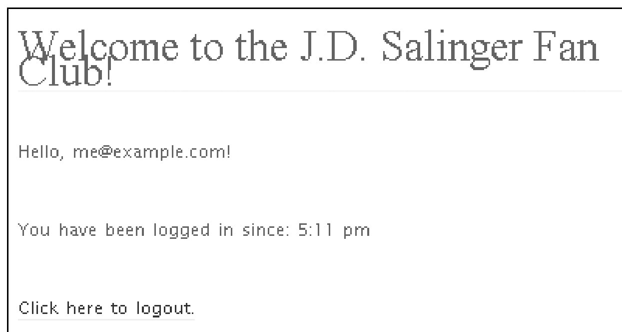


图9-19　在成功进行登录之后（在表单中使用*me@example.com*和*testpass*），用户被重新定向至本页，在这里将使用session的值对他们进行问候

✓提示

- 可以使用isset($_SESSION['var'])来查看特定的session值是否存在，就像用来检验其他的变量是否被设置那样。
- 在session中，数据都会以纯文本的形式保存在一个开放可读的文本文件中。请时刻注意session中保存的数据，永远不要将真实的敏感信息存在session中，如信用卡数据。
- 要提高安全性，可以将数据加密后再保存到session中，在读取session后再将其解密。这需要使用Mcrypt库和一些高级的PHP知识。

9.9　删除 session

知道如何删除session是非常重要的，就像知道如何删除cookie一样重要：有时候需要删除已经保存的session数据。session的数据在两个地方存在，因此需要在两个地方进行删除操作。但是首先必须从函数session_start()开始，通常像下面这样：

```
session_start();
```

然后，再次设置数组$_SESSION来删除session的值：

```
$_SESSION = array();
```

最后，需要从服务器上删除session数据（保存在临时文件中）。可以通过下面的方式进行操作：

```
session_destroy();
```

我们编写logout.php，它将用来删除session，以有效地注销用户。

⇒ 删除session

(1) 在文本编辑器或者IDE中新建一个PHP脚本，命名为logout.php（参看脚本9-8）。

```
<?php // 脚本9-8 - logout.php
```

脚本9-8　删除session需要这样的3个步骤：开启session，重新设置数组，然后删除session数据

```
1   <?php // 脚本9-8 - logout.php
2   /* 登出页面，会删除session数据。*/
3
4   // 开始session部分：
5   session_start();
6
7   // 删除session变量：
8   unset($_SESSION);
9
10  // 重置session数组：
11  $_SESSION = array();
12
13  // 定义页面标题并包含页头文件：
14  define('TITLE', 'Logout');
15  include('templates/header.html');
16
17  ?>
18
19  <h2>Welcome to the J.D. Salinger Fan Club!</h2>
20  <p>You are now logged out.</p>
21  <p>Thank you for using this site. We hope that you liked it.<br/>
22  Blah, blah, blah...
23  Blah, blah, blah...</p>
24
25  <?php include('templates/footer.html'); ?>
```

(2) 开始编写session部分：

```
session_start();
```

请记住，直到用这个函数激活session后才能对session进行删除。

(3)重置session数组

```
$_SESSION = array();
```

在第7章介绍过，array()函数会创建一个新的空数组。将这个函数调用的结果赋给$_SESSION，$_SESSION中所有的键-值对都会被删除。

(4) 删除服务器上的session数据：

```
session_destroy();
```

该步骤告知PHP将服务器上真正的session文件删除。

(5) 将HTML的header部分包含进来，并且完成PHP部分：

```
define('TITLE', 'Logout');
include('templates/header.html');
```

```
?>
```

(6) 完成页面的内容部分：

```
<h2>Welcome to the J.D. Salinger Fan Club!</h2>
<p>You are now logged out.</p>
<p>Thank you for using this site. We hope that you liked it.<br/>
Blah, blah, blah...
Blah, blah, blah...</p>
```

(7) 将HTML页脚包含进来：

```
<?php require ('templates/footer.html'); ?>
```

(8) 将文件保存为logout.php，放置在启用了PHP的服务器上适当的目录中，并且通过单击welcome.php上的链接来在Web浏览器上测试（参见图9-20）。

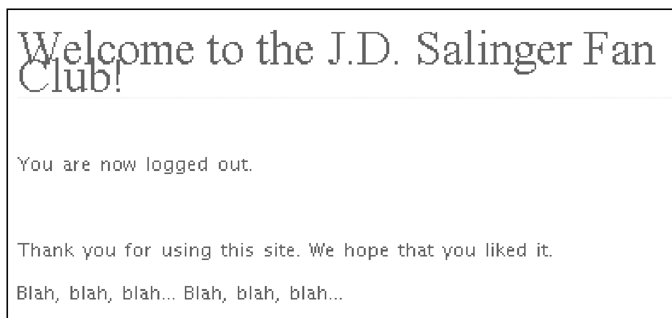


图9-20 注销页面删除了session数据

✓提示

- 可以使用unset($_SESSION['var'])来删除一个单独的session值。
- 服务器上的PHP模块将自动依照它配置中的设置执行垃圾回收。PHP假定这些session文件不再需要，使用垃圾回收以手动从服务器上删除它们。
- 当session ID必须附加给站点中每个链接（以便让每个页面都能够接受到session ID）的情况下，可以让PHP使用session而不使用cookie。如果启用了enable_trans_side设置（在php.ini文件中），PHP将为对此进行处理。

9.10 回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

9.10.1 回顾

- cookie在什么地方存储数据？session在什么地方存储数据？哪个更安全？

- 说出两个处理cookie问题的调试技术。
- `setcookie()`函数的两个参数`path`和`domain`是如何限制cookie访问方式的?
- 如何删除一个cookie?
- 如果要创建或访问session数据，页面必须调用哪个函数?
- 为什么session会使用cookie（默认情况下）?

9.10.2　实践

- 如果还未安装Firebug扩展包，在Firefox中安装它。前提是先要安装Firefox。
- 在PHP手册中查找`setcookie()`函数。了解关于操作cookie的其他说明信息和用户注释。
- 重写`customize.php`，以便让该脚本也能应用用户偏好。提示：应考虑到cookie在设置后不会立即生效。在表单提交之后，应该使用`$_GET`值编写CSS代码，第一次打开页面使用`$_COOKIE`（如果cookie已经存在），否则使用默认值。
- 将`customize.php`页面中的表单改为粘性表单，使其显示用户的当前选择。
- 重写`welcome.php`，在`print`语句中使用双引号打印出用户Email地址和欢迎信息。
- 这次加大难度，重写`welcome.php`，在`print`语句中使用双引号打印出用户的登录时长。提示：需要使用变量。
- 重写最后三个脚本，让session使用自定义名称。

第 10 章

创建函数

本章内容

- ❑ 创建和使用简单函数
- ❑ 创建和调用接受参数的函数
- ❑ 设置默认参数值
- ❑ 创建和使用带有返回值的函数
- ❑ 理解变量作用域
- ❑ 回顾和实践

在本书中，我们已经使用了几十个函数（如date()、setcookie()和number_format()），它们都提供了非常有用的功能。尽管这些函数都已经被PHP定义过，但是在这里我们将创建自己的函数。需要注意的是，函数创建之后，可以像使用PHP内置的函数那样使用它们。

创建函数能够为编程节省大量的时间。事实上，它们是创建Web应用程序和生成在将来的项目中使用的可靠的PHP代码库过程中重要的一步。

本章将介绍如何编写自己的函数以执行特定的任务。随后，将讲授如何向函数传递信息，在函数中使用默认的值，以及让函数返回值。此外，还将介绍函数和变量是如何共同工作的。

10.1 创建和使用简单函数

在编程时，你将会发现无论是在一段单独的脚本内还是在几段脚本的执行过程中，都会频繁地使用某些特定的代码段。将这些例程作为自定义的函数将节省大量时间，并且可以让编程过程更加简便，尤其是当Web站点变得越来越复杂时。如果创建一个函数，该函数就可以在每次被调用时发生相应的行为，类似于print在每次使用时都会向浏览器发送文本。

创建一个用户自定义函数的语法如下：

```
function function_name () {
        statement(s);
}
```

例如：

```
function whatever() {
       print 'whatever';
}
```

可以采取与为变量命名大致相同的规则来为函数命名，只是不需要以美元符号作为开头。其次，建议创建的函数名称是有意义的，就像应当编写有代表性意义的变量名称那样（create_header作为函数名将比function1要好得多）。请记住，不要使用空格（即不能使用两个分隔的词作为函数名称），这将会导致产生错误消息（可以使用下划线来替代空格）。不同于变量的是，PHP中的函数名不区分大小写，但是仍然需要坚持命名方法的一致性。

任何有效的PHP代码都能够在函数的*statement*(s)区域运行，包括对其他函数的调用。对于函数所包含的语句数量并没有限制，但是请确定每个语句的结尾都使用分号，就像在PHP脚本的其他部分那样。函数也可以包含任意控制结构的组合：条件或循环。

只要必需的要素都齐全，函数的格式并不是那么重要。这些要素包括function这个词、函数名称、打开和关闭的圆括号、打开和关闭的花括号以及语句。按照惯例，函数的语句通常都会相对于function关键字那行进行缩进，以便看起来更加明晰，就像在循环和条件中所作的那样。在任何情况下，你都要选择一种喜欢的格式样式（语法正确且合乎逻辑）并且坚持一直使用它。

可以通过引用的方式对函数进行调用，就像调用内置函数那样。下面这行代码：

```
whatever();
```

将会执行前面定义的函数语句部分中的print语句。

首先创建一个函数，用来生成表单中的月、日和年的下拉菜单（参见图10-1）。

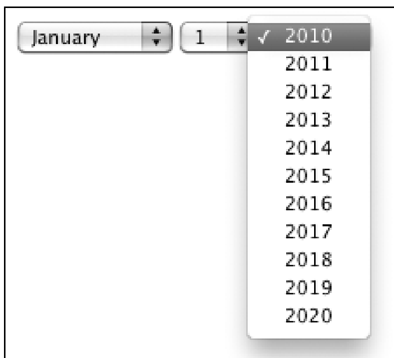


图10-1　这些下拉菜单是使用一个用户自定义函数所创建的。这项技术将会让代码更具移植性而不影响最终的结果

⇒ 创建和调用基本函数

(1) 在文本编辑器或者IDE中创建一个新的PHP文档，命名为menus.php（参看脚本10-1）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
```

```
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>Date Menus</title>
</head>
<body>
```

脚本10-1 脚本中定义的函数为表单创建了3个下拉菜单

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>Date Menus</title>
7   </head>
8   <body>
9   <?php // 脚本10-1 - menus.php
10  /* 脚本定义并调用函数。*/
11
12  // 函数创建3个下拉菜单:月、日、年。
13  function make_date_menus() {
14
15    // 将月以数组形式存储:
16    $months = array (1 => 'January', 'February', 'March', 'April', 'May', 'June',
      'July', 'August', 'September', 'October', 'November', 'December');
17
18    // 创建月下拉菜单:
19    print '<select name="month">';
20    foreach ($months as $key => $value) {
21      print "\n<option value=\"$key\">$value</option>";
22    }
23    print '</select>';
24
25    // 创建日下拉菜单:
26    print '<select name="day">';
27    for ($day = 1; $day <= 31; $day++) {
28      print "\n<option value=\"$day\">$day</option>";
29    }
30    print '</select>';
31
32    // 创建年下拉菜单:
33    print '<select name="year">';
34    $start_year = date('Y');
35    for ($y = $start_year; $y <= ($start_year + 10); $y++) {
36      print "\n<option value=\"$y\">$y</option>";
37    }
38    print '</select>';
39
40  } // 结束make_date_menus()函数。
41
42  // 创建表单:
43  print '<form action="" method="post">';
44  make_date_menus();
45  print '</form>';
46
```

```
47  ?>
48  </body>
49  </html>
```

(2) 开始PHP代码部分：

```
<?php // 脚本10-1 - menus.php
```

(3) 开始定义函数：

```
function make_date_menus() {
```

这个函数的名称为make_date_menus，它在描述函数作用的同时还非常易于记忆。

(4) 创建month下拉菜单：

```
$months = array (1 => 'January', 'February', 'March', 'April', 'May', 'June', 'July',
→'August', 'September', 'October','November', 'December');
print '<select name="month">';
foreach ($months as $key => $value) {
  print "\n<option value=\"$key\">$value</option>";
}
print '</select>';
```

生成一个月份的列表，首先要创建一个月份名称的数组，数值类型的索引从1开始，代表January。当为第一个数组元素指明索引时，其余的元素将会跟随延续索引编号，而不用分别指明。

在数组创建之后，将打印出初始select标签。然后，在数组$months上运行foreach循环。对于数组中的每一个元素，打印HTML option标签，并且以数组的键（从数字1到12）作为option的值，同时以数组的值（January到December）作为显示文本。每行都用一个换行符（\n）开头以便在HTML源代码中每个option都单独成行。

(5) 创建day下拉菜单：

```
print '<select name="day">';
for ($day = 1; $day <= 31; $day++) {
  print "\n<option value=\"$day\">$day</option>";
}
print '</select>';
```

创建day菜单是非常容易的。可以使用一个简单的从数字1~31的for循环来做到。

(6) 创建year下拉菜单：

```
print '<select name="year">';
$start_year = date ('Y');
for ($y = $start_year; $y <= ($start_year + 10); $y++) {
  print "\n<option value=\"$y\">$y </option>";
}
print '</select>';
```

为了创建year下拉菜单，将使用date()函数来获取当前的年份。然后用for循环为这个年份创建接下来下10个年份的选项。

(7) 结束函数：

```
} // 结束make_date_menus()函数。
```

当创建函数时，很容易会因为遗忘了关闭的花括号而产生解析错误。你可以通过添加注释来帮助记住这最后一步。

(8) 编写表单标签，并且调用函数：

```
print '<form action="" method="post">';
make_date_menus();
print '</form>';
```

print语句用来创建HTML表单标签。如果不这么做，date下拉菜单将不能在脚本中正确地显示出来。

函数创建完成之后，就可以通过函数名（不要把名字拼写错）调用它，注意在后面加上圆括号。

(9) 完成PHP和HTML代码：

```
?>
</body>
</html>
```

(10) 将文件保存为menus.php，放置在启用了PHP服务器上适当的目录中，并且在Web浏览器上运行（参见图10-1）。

(11) 你可以查看该页面的HTML源代码，看看哪些东西是动态生成的（参见图10-2）。

```
<form action="" method="post"><select name="month">
<option value="1">January</option>
<option value="2">February</option>
<option value="3">March</option>
<option value="4">April</option>
<option value="5">May</option>
<option value="6">June</option>
<option value="7">July</option>
<option value="8">August</option>
<option value="9">September</option>
<option value="10">October</option>
<option value="11">November</option>
<option value="12">December</option></select><select name="day">
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
```

图10-2 该页面源代码，显示出make_date_menus()函数中的print语句生成的HTML

✓提示

- 如果出现some_function...错误消息，这意味着你正在调用一个不存在的函数（参见图10-3）。这可能是由于函数名称拼写错误，也可能是由于PHP版本不支持该函数，请重新查看PHP手册。如果在调用用户自定义函数时看到这个错误，可能是函数没有定义或拼

写错误所导致的。重新检查函数的定义和调用部分的拼写，查看是否有错误发生。

```
Fatal error: Call to undefined function make_date_menu() in
/Users/larryullman/Sites/phpvqs4/menus.php on line 44
```

图10-3　这个错误的意思是PHP脚本没有找到指定名称的函数定义。在这个例子中，出错的原因是在函数调用时漏掉了“s”：正确的写法make_date_menus()与错误的写法make_date_menu()

- 函数function_exists()将根据PHP中是否存在某个函数而返回TRUE或者FALSE。这适用于用户自定义的函数和PHP内置的函数：

```
if (function_exists('some_function')) { ...
```

- 尽管在PHP中不需要在调用函数之前做预先定义，但是推荐的做法是保持在脚本的开头（或放在引入文件中）就定义好函数的习惯，最好不要在脚本的结尾处定义。
- 一些人喜欢在他们的函数中使用这样的语法：

```
function function_name()
{
  statement(s);
}
```

- 用户自定义的函数为PHP脚本增加了额外的内存需要，因此你应该恰当地使用它们。如果发现函数只不过调用了另外一个PHP函数或者只是一行代码，就可能没有很好地利用自定义函数。

10.2　创建和调用接受参数的函数

尽管创建一个简单的函数是有用的，但更好的作法是编写一个能够接受输入并且能够依照输入进行某些操作的函数。这些函数输入被称为*参数*（argument或者parameter）。这个概念之前已经看到过：sort()函数接受一个数组作为参数，而函数正是对这个数组进行排序操作的。

编写接受参数的函数的语法如下所示：

```
function function_name($arg1, $arg2, ...){
       statement(s);
}
```

函数的参数以变量的形式存在，而这些变量被赋予调用函数时向函数发送的值。这些变量使用同PHP中其他变量相同的命名规则进行定义：

```
function make_full_name($first, $last) {
       print $first . ' ' . $last;
}
```

调用接受输入（参数）的函数和调用不接受参数的函数非常类似——只需记住传递所需的值即可。这可以通过传递变量实现，如：

```
make_full_name($fn, $ln);
```

或者发送字符串实现，如：

```
make_full_name('Larry', 'Ullman');
```

或者结合上述两种方法，如：

```
make_full_name('Larry', $ln);
```

需要注意的是参数是被逐字传送的：在函数定义中的第一个变量将被赋予调用行中的第一个值，第二个函数变量被赋予第二个调用值，依此类推（参见图10-4）。函数并没有智能到能够直观地理解我们对于值之间关联的认识。如果值传递失败，函数将假定其值为空（值为空并不等于数值0，它是真实存在的值，更接近于词语“无”）。同样地，如果一个函数有4个参数而只传递了3个——第4个将为空，这也会产生错误（参见图10-5）。

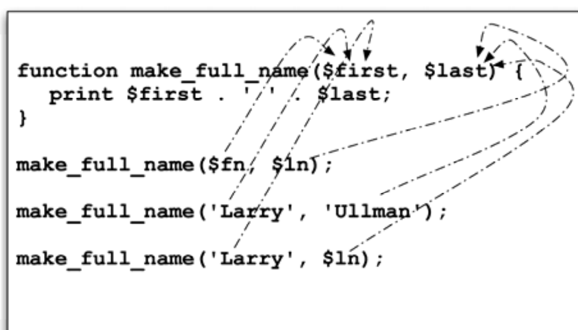


图10-4　函数调用中的值是如何赋给函数参数的

Warning: Missing argument 2 for make_text_input(), called in /Users/larryullman/Sites/phpvqs4/sticky1.php on line 38 and defined in **/Users/larryullman/Sites/phpvqs4/sticky1.php** on line **14**

图10-5　在调用（用户自定义或者PHP内置的）函数时，传递的参数个数不对将会导致出现错误消息

接下来创建一个更有意思的示例来诠释接受参数的函数。在第8章介绍了创建粘性表单，使其显示之前提交过的值。粘性文本框的代码如下：

```
First Name: <input type="text" name="first_name" size="20"
→value="<?php if (isset($_POST ['first_name'])) { print htmlspecialchars($_POST
→['first_name']); } ?>" />
```

由于页面中有很多表单都会反复使用相同的代码，例如第8章中的`register.php`因此这里非常适合使用用户自定义函数。

接下来的脚本将定义并调用一个函数，该函数创建了粘性文本框。函数中的一个参数用于接受文本框的名字，另一个参数用于接受文本框标签（文本提示）。这个函数会被脚本多次调用，以生成各种文本框（参见图10-6）。表单提交后，表单会记住之前输入的值。

图10-6 以上三个表单元素是由用户自定义函数创建的

⇒ 创建和调用接受参数的函数

(1) 在文本编辑器或者IDE中创建一个新的PHP文档，命名为`sticky1.php`（参看脚本10-2）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Sticky Text Inputs</title>
</head>
<body>
```

脚本10-2 `make_text_input()`函数接受两个参数，指明文本框的名字和标签

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>Sticky Text Inputs</title>
7   </head>
8   <body>
9   <?php // 脚本10-2 - sticky1.php
10  /* 脚本定义并调用函数，该函数创建粘性文本框。*/
11
12  // 函数创建粘性文本框。
13  // 函数接受两个参数。
14  function make_text_input($name, $label) {
15
16    // 打印段落标签和label标签：
17    print '<p><label>' . $label . ': ';
18
19    // 打印文本框：
20    print '<input type="text" name="' . $name . '" size="20" ';
21
22    // 加入文本框预设值：
23    if (isset($_POST[$name])) {
24      print ' value="' . htmlspecialchars($_POST[$name]) . '"';
25    }
26
```

```
27    // 完成文本框，关闭label和段落标签：
28    print ' /></label></p>';
29
30  } // 结束make_text_input()函数。
31
32  // 创建表单：
33  print '<form action="" method="post">';
34
35  // 创建文本框：
36  make_text_input('first_name', 'First Name');
37  make_text_input('last_name', 'Last Name');
38  make_text_input('email', 'Email Address');
39
40  print '<input type="submit" name="submit" value="Register!" /></form>';
41
42  ?>
43  </body>
44  </html>
```

(2) 开始PHP代码部分：

```
<?php // 脚本10-2 - sticky1.php
```

(3) 开始定义函数：

```
function make_text_input($name,$label) {
```

make_text_input()函数有两个参数，两个参数分别赋值给$name和$label变量。

(4) 打印段落标签和label标签：

```
print '<p><label>' . $label . ': ';
```

这个函数生成的代码与第8章给出的差不多，但是这里会将代码包括在段落标签中，文本框标签会放在label标签中。调用函数时，会将标签的值（如，First Name）传递到函数中。

(5) 打印文本框：

```
print '<input type="text"name="' . $name . '" size="20" ';
```

PHP的print语句用来创建HTML input标签，但标签name属性的值来自变量$name（调用函数时赋值，例如：first_name）。

(6) 如果需要，加入文本框预设值：

```
if (isset($_POST[$name])) {
  print ' value="' .
  vhtmlspecialchars($_POST [$name]) . '"';
}
```

第5步中的代码并没有结束文本框创建（没有使用/>关闭），因此还可以加入类似value="whatever"的语句，但这条语句只能在设置了$_POST['input_name']的情况下才可加入，所以这里加入了条件语句。同第8章的示例代码一样，元素的value属性值只能在运行完毕htmlspecialchars()函数时才可以打印。

(7) 完成文本框，关闭label和段落标签，结束函数：

```
print ' /></label></p>';
} // End of make_text_input()function.
```

(8) 创建form标签，调用函数：

```
print '<form action=""method="post">';
make_text_input('first_name','First Name');
```

表单必须使用POST方法，因为函数检查的是$_POST变量的值，这点很重要。如果表单会自动提交回相同的页面，可以忽略action值。

(9) 创建另外两个文本框：

```
make_text_input('last_name', 'Last Name');
make_text_input('email', 'Email Address');
```

注意，脚本以三种不同的方式使用了同一个函数三次，生成三个完全不同的文本框。

(10) 完成表单：

```
print '<input type="submit" name="submit" value="Register!" /></form>';
```

表单需要一个提交按钮，测试表单的粘性功能。

(11) 结束PHP和HTML：

```
?>
</body>
</html>
```

(12) 将文件保存为sticky1.php，放置在启用了PHP服务器上适当的目录中，并且在Web浏览器上测试（参见图10-6和图10-7）。

图10-7 表单元素显示用户之前输入的值

✓提示

- ❑ 可以定义任何数量的函数，而不用像本章示例中描述的那样，即每个脚本中只有一个函数。
- ❑ 函数能够接受的参数没有数量限制。
- ❑ 一旦已经像这样定义了自己的函数，就可以将它们放置在外部文件中，并且可以在需要访问这个函数的时候请求该文件。

10.3　设置参数默认值

PHP允许函数拥有默认的参数值。要做到这些，只用在函数定义中为参数赋值即可：

```
function greeting($who = 'world') {
    print "<p>Hello, $who!</p>";
}
```

这个函数将使用预设值，除非它获取到一个覆盖默认值的值。换句话说，通过为一个参数设定默认值，可以在调用函数时呈现特定的可选参数。如果希望为参数假定一个特定的值，那么需要为其设定默认值，但仍然允许传递其他值（参见图10-8）。

```
greeting();
greeting('Zoe');
```

Hello, world!

Hello, Zoe!

图10-8　用默认值调用没有任何参数的函数（第一个问候），调用带有提供一个参数的函数意味着该值将会被使用（第二个问候）

注意，这不是用户自定义函数（或一个默认的参数值）的好例子，示例中使用它只是为了方便理解。

应当总是在其他标准参数（它们没有默认值）之后编写默认参数。这是因为PHP按照从调用行获取的顺序直接为参数赋值。因此，不可能出现省略第一个参数的值但却包含第二个参数的情况。例如，假设此时有：

```
function calculate_total($qty, $price = 20.00, $tax = 0.06) {...
```

如果用代码行调用函数：

```
calculate_total(3, 0.07);
```

我们期望将$qty设置为3，将$price的值保持为20.00，并且将$tax的值更改为0.07，但这是有问题的。

最终的结果将是$qty被设置为3，$price为0.07，而$tax则保持为0.06（参见图10-9），这并不是所希望的输出结果。为了达到期望的结果，正确的方法是使用下面的代码：

```
calculate_total(3, 20.00, 0.07);
```

让我们重新运行make_text_input()函数以获取对设定参数默认值的完整认识。

```
function calculate_total($qty, $price = 20.00, $tax = 0.06) {
   // Make the calculations.
}
calculate_total(3, 0.07);
```

图10-9　根据函数参数赋值的方式，在调用函数时不能“跳过”某个参数

⇛ 编写一个使用默认值的函数

(1) 如果sticky1.php不在开启状态的话，在文本编辑器或者IDE中打开它（参看脚本10-3）。

(2) 为函数`make_text_input()`加入第三个参数及其默认值：

```
function make_text_input($name, $label, $size = 20) {
```

尽管我喜欢整洁的页面，习惯将所有的文本框做成同样的大小，但人的姓氏[①]和Emial地址通常比名字要长，因此最好可以调整文本框的大小。将文本框的大小作为参数，可以解决这个问题。文本框大小要有一个默认值，使其成为一个可选的参数。如果传入第三个参数，`$size`的默认值就会被这个参数的值取代。

脚本10-3　函数将接受两个参数，但其中只有一个是必需的。如果`$size`的值没有发送给函数，那么它的值将是20

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>Sticky Text Inputs</title>
7   </head>
8   <body>
9   <?php // 脚本10-3 - sticky2.php
10  /* 脚本定义并调用函数，该函数创建粘性文本框。*/
11
12  // 函数创建粘性文本框。
13  // 函数接受两个参数。
14  // 第三个参数是可选的（包含默认值）。
15  function make_text_input($name, $label, $size = 20) {
16
17    // 打印段落标签和label标签：
18    print '<p><label>' . $label . ': ';
19
20    // 打印文本框：
21    print '<input type="text" name="' . $name . '" size="' . $size . '" ';
22
23    // 加入文本框预设值：
24    if (isset($_POST[$name])) {
25      print ' value="' . htmlspecialchars ($_POST[$name]) . '"';
26    }
27
28    // 完成文本框，关闭label和段落标签：
29    print ' /></label></p>';
30
31  } // 结束make_text_input()函数。
32
33  // 创建表单：
34  print '<form action="" method="post">';
35
36  // 创建文本框：
37  make_text_input('first_name', 'First Name');
38  make_text_input('last_name', 'Last Name', 30);
```

① 这里指西方人的姓氏。——编者注

```
39  make_text_input('email', 'Email Address', 50);
40
41  print '<input type="submit" name="submit" value="Register!" /></form>';
42
43  ?>
44  </body>
45  </html>
```

(3) 改变文本框创建语句，加入$size变量：

```
print '<input type="text"name="' . $name . '"size="' . $size . '" ';
```

(4) 修改函数调用语句，改变文本框的大小：

```
make_text_input('first_name', 'First Name');
make_text_input('last_name', 'Last Name', 30);
make_text_input('email', 'Email Address', 50);
```

现在，第一个文本框会使用默认大小，而其他的文本框会更长一些。

(5) 将脚本保存为sticky2.php，放置在启用了PHP服务器上适当的目录中，并且在Web浏览器上测试（参见图10-10）。

First Name:	Peter
Last Name:	O'Toole
Email Address:	me@example.com
Register!	

图10-10 现在函数可以根据传入的参数改变大小，如果不传入参数，将使用默认大小（第一个文本框）

✓提示

- ❑ 可以使用一个空字符串（''）或者NULL（不带有引号）为函数的特殊参数传递空值。用其中任何一种方式都将覆盖原有的默认值。
- ❑ 正如第1章中提到的，PHP手册用方括号来标记函数参数。例如，当使用number_format()函数时，舍入到小数的位数是可选的：

```
string number_format(float number [, int decimals])
```

10.4 创建和使用带有返回值的函数

函数不仅可以接受参数，还能够返回值，这只需要两个步骤即可。首先，在函数中使用返回语句。第二，在调用函数的时候以某种方式使用输出。通常情况下，将会把返回的值赋给一个变量，但是还可以做更多操作，如直接打印出输出结果。这是接受两个参数并返回一个值的函数的基本格式：

```
function make_full_name($first, $last) {
  $name = $first . ' ' . $last;
  return $name;
}
```

可以这样使用函数：

```
$full_name = make_full_name($fn, $ln);
```

这里函数返回的值被赋给一个变量。它被立即打印出来：

```
print make_full_name($fn, $ln)
```

为了更好地诠释这个概念，我们创建一个函数，它将执行一个简单的计算并且对结果进行格式化。这段脚本将显示一个用户用于填写数量和价格的HTML表单（参见图10-11）。当表单被提交时（回到这个相同的页面），将打印出一个总值（参见图10-12）。

图10-11　这个简单的表单带有两个需要计算的值

图10-12　用户自定义函数计算的结果

⇛ 创建和使用带有返回值的函数

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为`calculator.php`（参看脚本10-4）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Cost Calculator</title>
</head>
<body>
```

脚本10-4　该脚本显示并处理了一个HTML表单以便执行一些基本的计算。脚本使用了一个接受两个参数和一个返回值的函数

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Cost Calculator</title>
7   </head>
```

```
8   <body>
9   <?php // 脚本10-4 - calculator.php
10  /* 脚本显示并处理HTML表单。
11  函数将数量乘以单价，以计算总值。*/
12
13  // 函数执行计算功能。
14  function calculate_total ($quantity, $price) {
15
16    $total = $quantity * $price; // 计算。
17    $total = number_format ($total, 2); // 格式化数字，保留两位小数。
18
19    return $total; // 返回值。
20
21  } // 结束函数。
22
23  // 检查表单是否提交：
24  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
25
26    // 检查表单数据：
27    if ( is_numeric($_POST['quantity']) AND is_numeric($_POST['price']) ) {
28
29      // 调用函数并打印结果：
30      $total = calculate_total($_POST['quantity'], $_POST['price']);
31      print "<p>Your total comes to $<span style=\"font-weight:
        bold;\">$total</span>.</p>";
32
33    } else { // 输入的数值不适宜。
34      print '<p style="color: red;">Please enter a valid quantity and price!</p>';
35    }
36
37  }
38  ?>
39  <form action="calculator.php" method="post">
40    <p>Quantity: <input type="text" name="quantity" size="3" /></p>
41    <p>Price: <input type="text" name="price" size="5" /></p>
42    <input type="submit" name="submit" value="Calculate!" />
43  </form>
44  </body>
45  </html>
```

(2) 开始PHP代码部分：

```
<?php // 脚本10-4 - calculator.php
```

(3) 定义函数：

```
function calculate_total ($quantity, $price) {
  $total = $quantity * $price;
  $total = number_format ($total, 2);
  return $total;
}
```

该函数接受两个参数：数量和单价，并且将它们进行乘法运算来创建总价。总价在被函数返回之前会被格式化。

尽管这样使用函数看上去显得有点笨拙，但是这里将一个单步运算放在函数中具有双重好处：首先，在脚本的开头处放置这个计算而不是将它隐藏在代码的其他地方，这会使日后在函数中查找和修改这个计算时变得更加容易；其次，可以不用复制代码而在脚本中重复这个动作。

(4) 开始条件部分以查看是否表单被提交：

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

因为该页面显示并处理了HTML表单，所以它用一个条件来检验页面的请求方式。如果是POST请求，就表明表单已经被提交。

(5) 验证表单数据并使用函数：

```
if ( is_numeric($_POST['quantity']) AND is_numeric($_POST['price']) ) {
  $total = calculate_total ($_POST['quantity'], $_POST['price']);
  print "<p>Your total comes to $<span style=\"font-weight:
  →bold;\">$total</span>.</p>";
```

这部分PHP代码（处理被提交的表单）首先检验是否已经填写数量和单价。如果是，总价的值将通过调用`calculate_total()`函数计算，并将结果赋给`$total`变量。然后会打印出结果。

(6) 完成条件语句：

```
  } else {
    print '<p style="color: red;">Please enter a valid quantity and price!</p>';
  }
}
```

如果某个表单变量没有被正确地提交，将会打印出一条消息来指明这个问题。最后的花括号结束表单的提交条件语句。

应用一小段CSS代码来打印消息（在这里和第(5)步中）。

(7) 显示HTML表单：

```
?>
<form action="calculator.php" method="post">
  <p>Quantity: <input type="text" name="quantity" size="3" /></p>
  <p>Price: <input type="text" name="price" size="5" /></p>
  <input type="submit" name="submit" value="Calculate!" />
</form>
```

表单本身很简单，它向用户请求两个不同的值（参见图10-8）。隐藏的表单输入作为处理代码的触发器，表明表单已提交。因为这个表单是在主提交条件之外创建的，因此它总是显示在页面上。

(8) 完成HTML页面：

```
</body>
</html>
```

(9) 保存页面为`calculator.php`，放置在启用了PHP的服务器上适当的目录中，并且在Web浏览器中进行测试（参见图10-12）。

返回多个值

用户自定义的函数常常只返回一个单独的值，但是通过使用数组，它也可以返回多个值。以下是这样操作的示例：

```
function some_function($a1, $a2) {
  // 做你想做的事情。
  return array($v1, $v2);
}
```

然后，调用这个函数，用`list()`函数将数组元素赋值给单独的变量：

```
list($var1, $var2) = some_function($p1,$p2);
```

最终结果是函数中的`$v1`被赋值给PHP脚本中的`$var1`，函数中的`$v2`被赋值给`$var2`。

✓**提示**

❑ 可以在一个函数中只执行一条返回语句，但是相同的函数能够有多条返回语句。作为示例，可以编写一个函数用来检验条件并用以指明条件是否被满足。在这种情况下，可以在函数中编写这样的代码：

```
if (condition) {
  return TRUE;
} else {
  return FALSE;
}
```

函数返回的结果不是TRUE就是FALSE，用以指示是否满足声明的条件。

10.5　理解变量作用域

在没有学习函数之前介绍*变量*，*作用域*没有任何意义。因此，之前我们并没有介绍变量作用域。现在，已经对函数有所了解，本节就将再次回顾关于变量的话题，并且讨论变量和函数是如何协同工作的一些细节。

正如在10.2节中看到的那样，可以用将变量作为参数的方式向函数发送变量。然而，也可以引用使用了全局声明函数中定义的外部变量。正因为有变量的作用域，才使这种应用成为可能。变量的作用域是它存在的领域。默认情况下，脚本中编写的变量存在于脚本的生命周期中。相反，诸如`$_SERVER['PHP_SELF']`的环境变量将会一直在服务器中存在。

尽管函数可以创建一个新级别的作用域，但函数变量（函数的参数如同在函数中定义的任何变量一样）只在那个函数中存在，并且不能从该函数之外访问（这就是说，它们是*局部变量*，拥有*局部作用域*）。同样地，默认情况下，函数外的变量不能在函数内使用（参见图10-13）。即使某个变量是函数调用的参数，也只是这个变量的值传给函数的，而不是变量本身的值。

`global`声明大致的意思是：希望这个函数内的变量能够指向函数外具有相同名称的变量。换句话说，`global`声明将一个拥有局部作用域的局部变量转变为拥有全局作用域的全局变量。当变量在函数之外时（也就是说假设函数被调用），在函数中对这个变量所作的任何变更都会被

传递给变量，而不必使用返回命令。

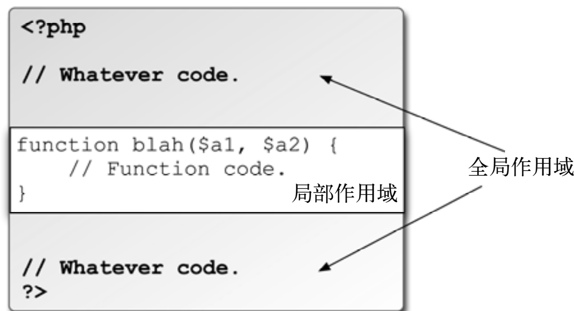


图10-13　在脚本中加入函数定义，就添加了另一个变量作用域

global声明的语法如下：

```
function function_name($args) {
       global $variable;
       statement(s);
}
```

这将导致另外一个关于函数和变量的问题：因为有变量作用域存在，函数中的局部变量是与在函数之外的变量不同的实体（也许有不同的值），即使这两个变量使用完全相同的名称。让我们进一步明确这一点……

假设有：

```
function test($arg) {
       // 做你想做的事情。
}
$var = 1;
test($var);
```

当函数被调用时，$var的值将被赋给$arg，因此它们的值是一样的，但是它们的名称不同，并且是不同的变量。但是，如果函数中的参数名称也是$var时：

```
function test($var) {
       // 做你想做的事情。
}
$var = 1;
test($var);
```

这样函数中的$var变量会被赋予同函数外原始$var相同的值，但是它们仍然是两个单独的变量。一个是在函数内拥有作用域，而另外一个的作用域是在函数外。这意味着在函数内和函数外使用名称完全相同的变量不会产生冲突。只是需要记住，它们是不同的变量即可。在函数内变量的值只对函数内的变量产生作用，示例如下（参见图10-14）。

```
function add_one($n) {
       $n++;
       print 'Added one!<br/>';
```

```
}
$n = 1;
print "\$n equals $n<br/>";
add_one($n);
print "\$n equals $n<br/>";
```

这些都是正确的，除非使用全局声明，当然，这将使这两个变量变得相同（参见图10-15）：

```
function add_one() {
       global $n; // 相同!
$n++;
       print 'Added one!<br/>';
}
$n = 1;
print "\$n equals $n<br/>";
add_one();
print "\$n equals $n<br/>";
```

注意，在这个例子中，不再需要将变量的值传递到函数中。

$n equals 1
Added one!
$n equals 1

图10-14　改变函数内局部变量的值不会影响同名的全局变量

$n equals 1
Added one!
$n equals 2

图10-15　改变函数内的全局变量将会影响函数外的变量

为了诠释变量的作用域，我们用global声明重写calculator.php脚本。

⇛ 使用global声明

(1) 如果calculator.php不是开启状态的话，在文本编辑器或者IDE中打开它（参看脚本10-4）。

(2) 在函数定义之前，添加下面的代码（参看脚本10-5）：

```
$tax = 8.75;
```

脚本10-5　脚本中的函数能够使用$tax变量（即便它并没有被传递给函数），这要归功于global声明


```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6     <title>Cost Calculator</title>
7   </head>
8   <body>
9   <?php // 脚本10-5 - calculator.php #2
10  /* 脚本显示并处理HTML表单。
11  函数将计算包含税率在内的总值。 */
```

```
12
13  // 定义税率：
14  $tax = 8.75;
15
16  // 函数执行计算功能。
17  function calculate_total ($quantity, $price) {
18
19    global $tax;
20
21    $total = $quantity * $price; // 计算。
22    $taxrate = ($tax / 100) + 1;
23    $total = $total * $taxrate; // 纳入税率后的总值。
24    $total = number_format ($total, 2); // 格式化数字，保留两位小数。
25
26    return $total; // 返回值。
27
28  } // 结束函数。
29
30  // 检查表单是否提交：
31  if (isset($_POST['submitted'])) {
32
33    // 检查表单数据：
34    if ( is_numeric($_POST['quantity']) AND is_numeric($_POST['price']) ) {
35
36      // 调用函数并打印结果：
37      $total = calculate_total($_POST['quantity'], $_POST['price']);
38      print "<p>Your total comes to $<span style=\"font-weight: bold;\">$total
        </span>, including the $tax percent tax rate.</p>";
39
40    } else { // 输入的数值不适宜。
41      print '<p style="color: red;">Please enter a valid quantity and price!</p>';
42    }
43
44  }
45  ?>
46  <form action="calculator.php" method="post">
47    <p>Quantity: <input type="text" name="quantity" size="3" /></p>
48    <p>Price: <input type="text" name="price" size="5" /></p>
49    <input type="submit" name="submit" value="Calculate!" />
50    <input type="hidden" name="submitted" value="true" />
51  </form>
52  </body>
53  </html>
```

创建一个`$tax`变量，使它拥有将在费用计算中用到的一系列的值。它被赋予函数之外的一个值，这是因为在这之后它将会在脚本的主体部分被用到。

(3) 在函数定义部分，添加一个全局声明：

```
global $tax;
```

这个声明告诉函数要将相同的`$tax`变量作为存在于函数之外的变量。

(4) 在函数中的`$total`被格式化之前，重新用税率计算值：

```
$taxrate = ($tax / 100) + 1;
$total = $total * $taxrate;
```

为了将税值加到总值中，需要先将税除以100，以获得一个百分比。然后为这个值加上1以获得一个乘数。将这个结果同先前的总值相乘以获得新的总值，即为最后的总值。

请注意，是用一个`$taxrate`变量（基于`$tax`）来执行这个计算的。这是因为在这之后将打印出`$tax`的值，并且在这里对之所作的任何更改都能反映出来（因为它是一个全局变量）。

(5) 在主`print`代码行（在函数调用之后）添加下面的代码以便打印出税率：

```
print "<p>Your total comes to $<span style=\"font-weight:
→bold;\">$total</span>, including the $tax percent tax rate.</p>";
```

在脚本的开头部分被定义的变量`$tax`最终在结尾处被打印。如果还没有使用函数内的`$taxrate`变量，并且替换为`$tax`这个全局变量，对计算结果的影响将通过打印值的方式表现出来。

(6) 保存脚本，放置在启用了PHP服务器上适当的目录中，并在Web浏览器中进行测试（参见图10-16和图10-17）。

图10-16　再次运行表单

图10-17　计算过程现在使用了`$tax`这个全局变量

✓提示

- ❑ 常量和超全局数组（`$_GET`、`$_POST`、`$_COOKIE`和`$_SESSION`）能带来额外的益处，它们通常不需全局声明就能在函数内使用（正因如此，它们被称为*超全局*）。
- ❑ 每个函数都有它自己的、独立的局部作用域。

函数设计理论

理解自定义函数的语法很重要，但你还需要深入地理解函数设计理论。恰当的用户自定义函数可以很容易地重用，重用的机会也会较多（如果网页只是偶尔需要调用某个函数，就不必使用自定义函数）。函数设计中有一个“黑盒”思想：程序员不必了解函数的内部结构就可以恰当地使用函数。PHP中的函数就是这一思想的示例：你并不清楚这些函数背后的实现代码，但你却可以充分发挥它们的功能。

为了支持“黑盒”理论，设计函数时要注意的一点是在使用全局变量时必须非常谨慎。可以说，应该向一个函数传递它所需要的所有信息，而不必使用全局变量（包括超级全局变量和常量）。

函数设计时也不能建立在假设的基础之上，例如，make_text_input()假设表单的提交方式是POST，这是不可取的。

在设计函数时，既不要依赖全局变量，也不要假设任何函数的外部条件为真，这样函数才能更为独立且移植性更好——简言之，更棒。

10.6　回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

10.6.1　回顾

- 用户自定义函数的基本语法是什么？
- 用户自定义函数的命名规则是什么？
- 用户自定义函数的参数命名规则是什么？
- 如何为自定义函数设置参数默认值?
- 在10.5节的示例代码中要使用\$n？如果不加反斜杠会发生什么？
- 什么是变量作用域？函数中参数变量的作用域是什么？
- 包含文件中变量作用域是什么？注意：这个问题比较难！

10.6.2　实践

- 设置menus.php中的函数带有两个参数，一个参数表示起始年，另一个参数表示生成的总年数。第二个参数有一个默认值。重写函数体，在year的for循环中使用这些参数值。
- 重写make_text_input()函数，以便可以告诉它是在$_POST中还是在$_GET中查询已存在的值。
- 在make_text_input()函数基础上创建一个新函数，根据函数调用方式，创建一个文本框和一个密码框。
- 自己设计一个脚本，脚本中要使用自定义函数。

第 11 章

文件和目录

11

本章内容

- ❑ 文件权限
- ❑ 写入文件
- ❑ 锁定文件
- ❑ 读取文件
- ❑ 处理文件上传
- ❑ 导航目录
- ❑ 创建目录
- ❑ 增量读取文件
- ❑ 回顾和实践

要使Web应用程序真正做到更上一层楼，需要掌握存储和检索数据的方法。使用PHP有两种主要的用于存储数据的方法：使用文件（和目录）或数据库。本章讨论前者，而第12章讨论后者。理解这两种方法值得花一些时间。尽管数据库可以比基于文件的系统更强大也更安全，但是通过读写服务器上的简单文本文件来发送和检索信息，能够做到的事情也会令人很惊讶！

本章将学到有关文件权限的知识，然后了解写入、读取和锁定文件。在这之后，你会看到如何使用PHP处理文件上传、如何创建目录，以及从文件中读取数据的另一种方法。最后的两个示例还演示了一个简单的基于文件的注册和登录系统，你可以在自己的Web应用程序中使用该系统。

11.1 文件权限

在尝试写入和读取文件之前，必须理解文件权限。这个主题的范围非常广泛，读者也许希望能够持续深入讨论，但这里的讨论只是一个开始。首先本书要说明的是，这里提到的很多信息只会影响非Windows用户。根据我的经验，当在Windows计算机上运行PHP时，这些预备步骤并不是必需的。然而，从整体上理解权限是件好事，尤其是今后可能会在非Windows服务器上运行PHP脚本。

权限指定了在一个文件或目录上，用户可以做些什么。其选项包括*读取*、*写入*和*执行*（实际上，文件可以指定为*可执行*，而目录是*可搜索的*）。在此之上，这些选项可以分别设置给3种类型

的用户：

- 文件的所有者（创建文件或将其放置到服务器上的人）；
- 某特定组的成员，由服务器管理员设置，并且包括文件所有者；
- 其他用户（不属于前两个分类的用户）。

还有一个隐含的everyone级别，它包括前面提到的所有用户。

举个例子，如果使用FTP将文件上传到服务器，连接服务器的账户就是文件的所有者。文件的默认权限是所有人都可以读取，但只有所有者可以修改文件（即写入）。

问题是，在大多数情况下，PHP脚本得通过Web服务器应用程序运行，并被视为一个不同的服务器用户。因此，PHP和Web服务器能够读取上传的文件（从而让Web浏览器可以浏览这些文件），但在默认情况下，PHP无法修改这些文件。

对于本章中的示例，PHP需要能够对一些文件和目录进行写入。也就是说，你必须知道如何修改文件和目录的权限。也就是说让文件或目录可写（减少权限限制），这可能会带来安全问题，因此只有在绝对必需的时候才能这样做。

最后要解释一个人们普遍混淆的概念——“用户”究竟指的是什么？用户是在计算机上创建的账户。在你自己的计算机上，可能只有一个用户（你自己）或几个用户。服务器上一般会有多个用户，但是很多用户账户并不与登录服务器的真实用户相对应，而是与运行在服务器上的进程相对应。例如，服务器上可能有一个用户进程用来处理所有Web请求，还有另一个用户运行数据库应用程序。最重要的是要知道，“用户”不是指使用这台计算机的人，还要知道everyone指的是“服务器上的everyone”。如果你在服务器上创建了一个对所有用户可写的文件或目录，并不表示在Internet上的所有人都可以写这个文件或目录。总之，用户必须是服务器中认可的账户。

Web根目录

当讨论文件、目录和安全时，Web根目录是一个非常重要的概念。要掌握这个概念，首先要知道Web服务器中的文件有两种访问方式。第一种方式是访问自己计算机的文件系统，比如：`C:\inetpub\wwwroot\filename.php`。第二种方式是访问远程服务器，比如`/var/web/sitename/htdocs/filename.php`（该路径应该适用于*.nix系统）。

其次，存放在Web服务器特定目录中的文件也可以通过HTTP访问，例如，http://www.example.com/filename.php或http://localhost/filename.php。

了解以上概念之后，Web根目录的概念就是基础URL（比如：www.example.com）指向的文件系统中的文件夹。如果不做任何限制，Web浏览器可以访问Web根目录及其子目录的所有文件。但是Web浏览器不能访问Web根目录外面的文件。

在创建可写文件和目录时，将它们放在Web根目录之外会更安全。换句话说，如果Web页面放置在`C:\inetpub\wwwroot`或`/Users/`*`username`*`/Sites`中，此时，如果你将项目放置在`C:\inetpub`或`/Users/`*`username`*中，这些项目可以被本地运行的PHP访问到，但不能通过Internet访问。本章中的示例都遵从这种结构，你也应该严格遵守。

总之，保证目录的安全比保证文件的安全更重要。

11.1.1　创建文本文件

在第一个示例中，你将操作服务器上名为quotes.txt的文本文件。如果PHP脚本不具备操作该文件所需的正确权限，将会看到类似图11-1所示的错误消息。在继续进行本章内容之前，应该在服务器上创建quotes.txt并为其建立权限。

Warning: fopen(../quotes.txt) [function.fopen]: failed to open stream: Permission denied in **/Users/larryullman/Sites/phpvqs4/add_quote.php** on line **20**

图11-1　尝试对一个文件执行服务器所不允许的操作会导致出现“…failed to open stream: Permission denied…”消息。在这里服务器禁止使用fopen()函数尝试以写入方式来打开quotes.txt

⇛ 创建quotes.txt

(1) 打开文本编辑器或IDE并新建一个空白文档。

(2) 不要向文件中输入任何内容，将其保存为quotes.txt。

(3) 将文件移动到支持PHP的服务器上的Web根目录之外（参见图11-2）。

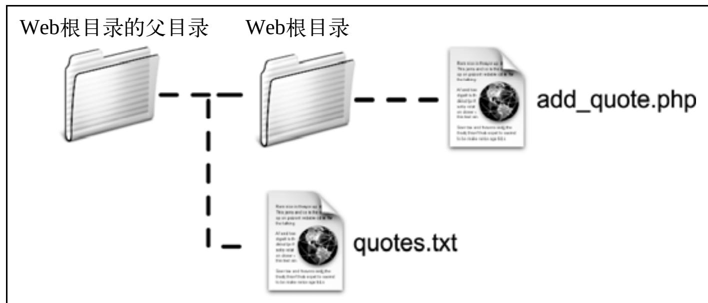


图11-2　理想情况下，quotes.txt文件应该放置在和Web文档目录同级的目录中（也就是说，不在Web文档目录之中）

框注“Web根目录”解释了应该将文件放置到相对于Web目录的什么位置，并解释了为什么要这样做。

✓提示

- ❑ 如果可以在服务器上找到提交的文件名，则file_exists()函数会返回TRUE。这可以用于在进行任何操作之前测试文件是否存在。

```
if (file_exists('somefile.ext')) { ...
```

- ❑ 假设PHP具备一个目录上的写权限，则可以使用PHP在该目录中创建一个空白文档。这是通过使用touch()函数来完成的：

```
touch('somefile.ext');
```

11.1.2　设置文件权限

前面的步骤顺序似乎看起来有些奇怪，但要为一个文件设置权限，该文件首先必须是存在的。不过只需要一个空的文件，因为稍后会使用PHP向其中写入数据。

该示例的最终预期结果是为任何人授予在`quotes.txt`上的读和写（但不包括执行）权限。如何做到这一点取决于：

- 是在本地计算机还是在远程服务器上运行PHP；
- 支持PHP的计算机所使用的操作系统。

遗憾的是，指出每个用户在各种环境下设置权限的步骤是不可能的，但这里给出了一些可用的概略指导原则和步骤。

⇒ 在远程服务器上设置文件权限

- 很多ISP会为用户提供一个基于Web的控制面板，可以用于设置文件权限（参见图11-3）以及其他主机参数。

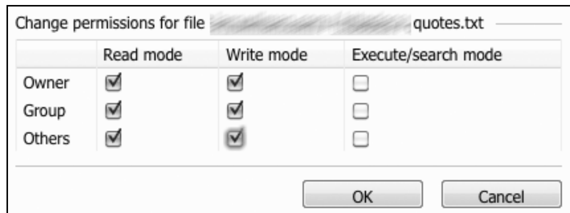


图11-3　该控制面板由主机公司提供，可以用来调整文件权限

- 也可以使用FTP客户端来修改文件的权限（参见图11-4）。

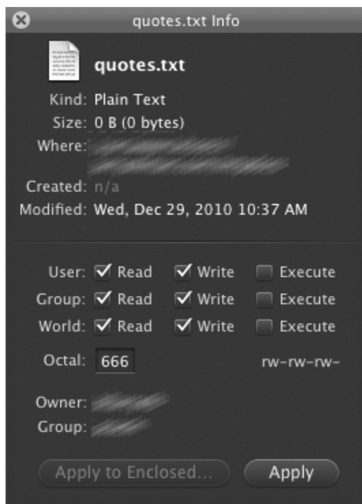


图11-4　TransmitFTP应用程序使用弹出窗口来设置文件权限

⇒ 在本地计算机上设置文件权限

- 如果使用自己的Windows计算机，则可能无需修改权限。要验证这一点，可以首先尝试一下每一个示例。如果PHP不能向文件或目录中写入数据，则需要使用下面的建议修改权限。
- Windows用户可以通过查看文件或目录的属性来修改权限。具体的对话框对于不同版本的Windows来说是不一样的，但需要调整的只是谁可以访问该文件以及如何访问。
- Mac OS X用户必须在Finder中选中文件，并从File菜单中选择Get Info。在这里，使用Ownership & Permissions子面板来调整文件的权限（参见图11-5）。
- 在Unix上（包括Linux和Max OS X用户），假如具有相应的权限，则可以在终端窗口中使用`chmod 0666 quotes.txt`命令行。

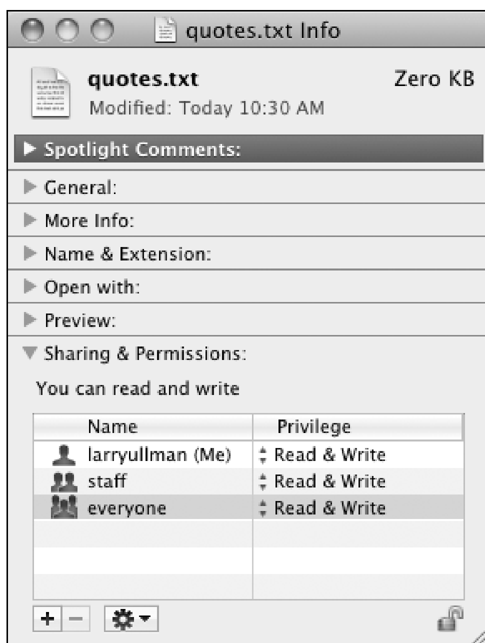


图11-5　Mac OS X的Get Info面板可以用于调整文件的所有权和权限，以及其他一些东西

✓提示

- 很多操作系统并没有PHP用户。PHP用户其实就是运行Web服务器应用程序（如Apache或IIS）的用户。在Unix家族中，Apache通常作为nobody运行。在Windows上，Web服务器经常以当前登录用户身份运行（该用户可能能够创建文件），这意味着无需修改文件权限。
- 如果对Telnet和`chmod`很熟悉，应该能够理解数值`0666`的含义。但这里还是要解释一下，因为有些读者并不熟悉。`0`只是一个前缀，指出该数值是以八进制格式书写的。每个`6`都是由读（`4`）和写（`2`）权限相加得到的——第一个`6`赋给文件或目录的所有者，然后是组，最后是其他用户。相对应，`0777`允许各种类型的用户具备读（`4`）、写（`2`）和执行（`1`）

权限。该数字适用于Unix系列的操作系统（Linux、Solaris和Mac OS X）。

- PHP具有很多用于修改文件或目录权限的函数，包括chgrp()、chown()和chmod()。然而，只有当PHP已经具备了修改文件或目录的权限后，这些函数才能工作。
- 与运行在专门的服务器上相比，运行在共享托管环境中的Web网站面临更多的安全问题，因为这种环境中的用户数量很多。举例来说，如果你创建了一个所有用户都可写的文件或目录，那么所有可以访问该服务器的用户都可以修改它（前提是这些用户知道有这么一个文件或目录，而且也没有其他权限限制）。

11.2　写入文件

要从文件中读取内容，首先需要向其中写入一些内容，因此本章首先介绍写入文件。写入文件最简单的方式是使用file_put_contents()函数：

```
file_put_contents($file, $data);
```

这个函数是PHP 5中添加的，它会打开文件并写入数据。第一个参数是文件名。可以使用绝对路径，也可以使用相对路径（参见框注“文件路径”）。第二个参数是要写入的数据，可以是字符串、数字或数组（只能是一维数组，不能使用多维数组）。无论是文件还是数据都不强制使用变量，但使用变量是常用的方法。

如果文件不存在，函数会尝试创建它。如果文件存在，文件的内容会被新的数据取代。如果需要将新内容追加到文件中，加入FILE_APPEND常量作为第三个参数：

```
file_put_contents($file, $data, FILE_APPEND);
```

当向文件中追加数据时，通常需要每块数据独占一行，因此每次提交时都应该以一个运行PHP的计算机操作系统的换行符结束。换行符可以是：

- \n——Unix和Mac OS X；
- \r\n——Windows。

这里需要加上双引号以使转义序列正常工作。

可以使用PHP专用的常量PHP_EOL，表示当前操作系统的换行符（即，\n或\r\n）：

```
file_put_contents($file,$data . PHP_EOL, FILE_APPEND);
```

如果使用的不是PHP 5或以上版本，将无法使用file_put_contents()函数，那么你就必须使用旧方法写入文件：首先，打开文件；其次，写入数据；最后，关闭文件。

```
$fp = fopen($file, mode);
fwrite($fp, $data . PHP_EOL);
fclose($fp);
```

要写入一个文件，必须在打开它时创建文件指针。PHP使用fopen()函数返回的文件指针来引用这个打开的文件。

打开文件时，需要重点考虑的是所使用的模式（mode）。模式指定了打开文件的方式，这取决于你希望对文件进行的操作。最为宽松的模式是a+，它允许读取或写入一个文件。如果文件不

存在则会创建文件，并且自动将新的数据追加到文件的末尾（因此是a）。反之，r只允许从文件中读取。表11-1列出了所有可能的模式。每种模式都可以追加一个b标志，强制以二进制模式打开文件。对于需要在多种操作系统上读取的文件来说，这是一种较为安全的选择。

表11-1　**fopen()**模式

模　　式	含　　义
r	只读；从文件的起始位置开始读取
r+	读或写；从文件的起始位置开始
w	只写；如果文件不存在则创建文件，并且覆盖现有内容
w+	读或写；如果文件不存在则创建文件，并且在写入时覆盖现有内容
a	只写；如果文件不存在则创建文件，将新数据追加到文件末尾（保留任何现有数据并向其中添加新数据）
a+	读或写，如果文件不存在则创建文件，将新数据追加到文件末尾（写入时）
x	只写；如果文件不存在则创建文件，如果文件存在则什么也不做（并发出一个警告）
x+	读或写；如果文件不存在则创建文件，如果文件存在则什么也不做（并在写入时发出一个警告）

fwrite()函数根据选中的模式向文件中写入新数据（作为函数调用的第二个参数进行传递）。

写入过程的最后一步，是再次使用文件指针来调用fclose()函数以关闭文件：

```
fclose($fp);
```

下面我们来创建一个表单，将用户提交的文本保存为纯文本文件（参见图11-6）。在本章稍后的部分中，还会创建其他PHP脚本来检索并随机显示这些文本。

在进入代码讲解之前，还有一个需要介绍的函数——is_writable()。这个函数返回布尔值，表示指定文件是否可被写入：

```
if (is_writable($file)) {...
```

在向文件（或目录）中写入数据之前，调用这个函数，可以避免权限错误。

图11-6　这个简单的表单允许用户提交一段文本并写入到文本文件中

文件路径

引用计算机上的任何文件或目录都有两种方式：使用绝对（absolute）或相对（relative）路径。绝对路径从计算机的根目录开始：

- `C:\somedir\somefile.txt`（Windows）；
- `/Users/username/somefile.txt`（Mac OS X）。

相对路径不会从计算机的根目录（`C:\`或`/`）开始，而是相对于当前的工作目录：

- `fileA.txt`（当前目录）；
- `./fileA.txt`（当前目录）；
- `dirB/fileB.txt`（dirB目录中）；
- `../fileC.txt`（父目录中）；
- `../dirD/fileD.txt`（同级目录中）。

将两个圆点放在一起可以表示当前目录的父目录。一个单独的圆点表示当前目录自身。（在Unix、Linux和Mac OS X上）如果一个文件的名字由一个圆点开始，则该文件是隐藏的。

从技术角度讲，只要保证引用的精确性，那么使用相对路径还是使用绝对路径来引用一个文件是无所谓的。初学者会觉得绝对路径比较好理解，但只能用在当前计算机上。相对路径可能不太好理解，但它可以保证网站移植时不出现路径错误。

⇛ 写入一个外部文件

(1) 在文本编辑器或IDE中新建一个PHP文档，命名为add_quote.php（参看脚本11-1）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Add A Quotation</title>
</head>
<body>
```

脚本11-1　该脚本用于获取用户提交的文本并将其存储到文本文件中

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Add A Quotation</title>
7   </head>
8   <body>
9   <?php // 脚本11-1 - add_quote.php
10  /* 脚本显示并处理HTML表单。脚本从文本框中获取文本并将其存储到文本文件中。 */
11
12  // 标识要使用的文件名：
13  $file = '../quotes.txt';
14
15  // 检查表单是否提交：
16  if ($_SERVER['REQUEST_METHOD'] == 'POST') { // 处理表单。
17
18    if ( !empty($_POST['quote']) && ($_POST['quote'] != 'Enter your quotation
```

```
      here.') ) { // 检查输入的文本。
19
20        if (is_writable($file)) { // 确认文件可以写入数据。
21
22           file_put_contents($file, $_POST['quote'] . PHP_EOL, FILE_APPEND);
             // 写入数据。
23
24           // 打印一条消息:
25           print '<p>Your quotation has been stored.</p>';
26
27        } else { // 无法打开文件。
28           print '<p style="color: red;">Your quotation could not be stored due to a
             system error.</p>';
29        }
30
31     } else { // 没有输入文本。
32         print '<p style="color: red;">Please enter a quotation!</p>';
33     }
34
35  } // 结束提交条件语句。
36
37   // 完成PHP节并显示表单:
38  ?>
39
40  <form action="add_quote.php" method="post">
41     <textarea name="quote" rows="5" cols="30">Enter your quotation here.</textarea>
       <br/>
42     <input type="submit" name="submit" value="Add This Quote!" />
43  </form>
44
45  </body>
46  </html>
```

(2) 创建PHP代码片段：

```
<?php // 脚本11-1 - add_quote.php
$file = '../quotes.txt';
```

由于这个脚本会引用同一个文件两次，因此最好将这个文件用一个变量标识。这样今后如果需要改变文件名或文件地址，就只需要修改一行代码。

标识的文件名为quotes.txt，这个文件应该位于该脚本的上一级目录中（比如，在Web根目录中，参见图11-2）。参见框注“文件路径”了解更多相关语法。

(3) 查看表单是否已经被提交：

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

该页面同时显示和处理HTML表单。这个条件语句检查表单是否已经被提交，如果已提交，则需要将文本写入到文本文件中。

(4) 检查输入的文本：

```
if ( !empty($_POST['quote']) && ($_POST['quote'] != 'Enter your quotation here.') ) {
```

这个简单的条件语句验证了用户提供的数据。第一部分确保了$_POST['quote']变量不为空，第二部分确保了该变量仍然没有默认值（如图11-6所示）。

(5) 确认文件可以写入数据：

```
if (is_writable($file)) {
```

将条件语句放置在一个条件语句中，通过这种方式可以令PHP脚本只在文件能够成功打开时尝试向其中写入内容。

(6) 将数据写入到文件，关闭文件，然后打印一条消息：

```
file_put_contents($file, $_POST['quote'] . PHP_EOL, FILE_APPEND);
print '<p>Your quotation has been stored.</p>';
```

第一行代码将用户提交的数据写入到文件之中。在写入数据中加入了PHP_EOL常量，因此每次提交的数据都会独占一行。

如果使用的不是PHP 5或以上版本，可以使用以下代码代替：

```
$fp = fopen ($file, 'ab')
fwrite($fp, $_POST['quote'] . PHP_EOL);
fclose($fp);
```

(7) 完成条件语句：

```
    } else { // 无法打开文件。
      print '<p style="color: red;">Your quotation could not be stored due
      →to a system error.</p>';
    }
  } else { // 没有输入文本。
    print '<p style="color: red;">Please enter a quotation!</p>';
  }
} // 结束提交条件语句。
```

第一个else检查PHP是否打开了用于写入文件的条件语句（参见图11-7）。如果看到了这个消息，则很可能是由权限引发的问题或文件引用不正确。第二个else完成了条件语句，检查文本是否已输入（参见图11-8）。最后的关闭花括号标志着整个提交条件语句的结尾。

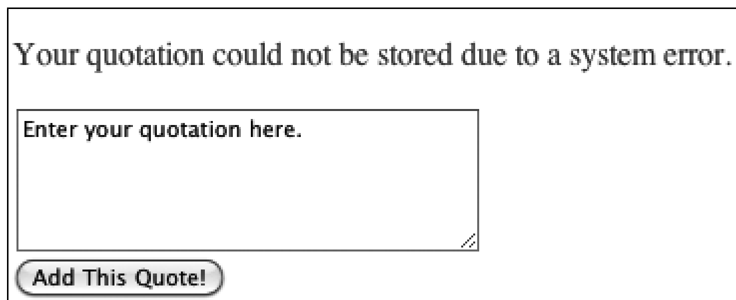


图11-7　如果PHP脚本无法找到quotes.txt文件，或该文件不可写，用户会看到这个信息

图11-8 脚本包括基本的表单验证

由于该页面处理了表单之后，又再一次显示了它（以便用户能继续输入文本），因此并没有将该表单作为else语句的一部分显示出来，这和本书中的其他示例是一样的。

(8) 完成PHP节：

```
?>
```

因为该脚本的其余部分都是标准的HTML，所以在这里通过一个PHP关闭标签来退出PHP代码段。

(9) 创建HTML表单：

```
<form action="add_quote.php" method="post">
  <textarea name="quote" rows="5" cols="30">Enter your quotation here.</textarea><br/>
  <input type="submit" name="submit" value="Add This Quote!" />
  <input type="hidden" name="submitted" value="true" />
</form>
```

该HTML表单显示了一个文本框，用于输入文本。该文本框具有预设值Enter your quotation here.，这是通过将文字放置到textarea标签中实现的。

(10) 完成HTML页面：

```
</body>
</html>
```

(11) 将文件保存为add_quote.php并将其放置在支持PHP的服务器上的适当目录中。

回顾图11-2，参考如何在服务器上放置add_quote.php和quotes.txt，以及它们之间的相对位置。如果不可能使用这种方式，或者觉得这太混乱了，也可以将两个文档放到同一个目录中（能够执行PHP脚本的那个目录），然后将$file一行修改为：

```
$file = 'quotes.txt';
```


(12) 在Web浏览器中多次运行该脚本（参见图11-9和图11-10）。

(13) 如果愿意的话，在文本编辑器中打开quotes.txt文件，确认数据都被写入了。

✓提示

- 注意所有与文件和目录相关的函数都只能在运行PHP的计算机（即服务器）上使用。服务器上的PHP脚本无法访问客户端的文件（只有将文件上传到服务器才可以访问）。

- 如果在运行该脚本时接收到了权限错误消息（参见图11-1），或者是没有正确地设置权限，或者是PHP脚本无法访问到数据文件。这可能是由于文件名拼写错误或没有正确引用文件在服务器上的路径。
- 如果正在运行的PHP版本是安全模式的，或设置了`open_basedir`指令，则在使用PHP访问文件和目录时会受到一些限制。请使用`phpinfo()`函数来查看服务器上的这些设置。

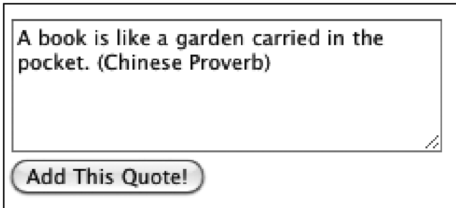


图11-9　填写表单

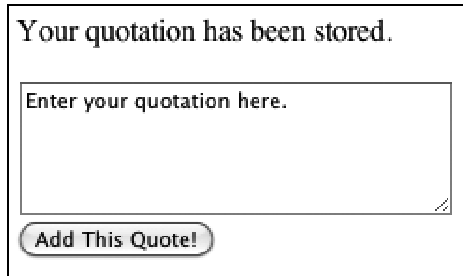


图11-10　如果一切正常将会看到该结果

11.3　锁定文件

尽管上一个示例可以运行得很好（希望如此），但它还可以进一步改进。如果同一时刻只有一个单独的用户提交表单，则不会有任何问题。但如果两个或更多用户在同一时刻提交不同的文本呢？当多个PHP脚本试图在同一时刻写入同一个文件时，就会出现问题（文件会损坏）。解决方法是在PHP对文件进行写入时，暂时锁定该文件。如果运行的是PHP 5.1及以上版本，可以在`file_put_contents()`函数中加入第三个参数`LOCK_EX`常量：

```
file_put_contents($file, $data, LOCK_EX);
```

如果要同时使用`LOCK_EX`和`FILE_APPEND`常量，可以使用二元OR运算符（`|`）：

```
file_put_contents($file, $data, FILE_APPEND | LOCK_EX);
```

两个变量的使用顺序无关紧要。如果使用的是早期的PHP版本（必须使用`fopen()`、`fwrite()`和`fclose()`），就需要在打开文件之后调用`flock()`函数：

```
$fp = fopen($file, 'a+b');
flock($fp, locktype)
```

各种不同的锁定类型由表11-2中列出的常量来指定。当脚本在处理文件时，需要将文件解锁。

表11-2　**`flock()`**锁定类型

锁	含　义
LOCK_SH	用于读取的共享锁
LOCK_EX	用于写入的独享锁
LOCK_UN	释放一个锁
LOCK_NB	非阻塞锁

例如，要在写入时临时锁定一个文件，可以使用下面的代码：

```
$fp = fopen($file, 'a+b');
flock ($fp, LOCK_EX);
fwrite ($fp, $data);
flock ($fp, LOCK_UN);
```

作为演示，下面我们将重写add_quote.php，在写入过程中锁定文件。

⇛使用文件锁

(1) 如果尚未打开add_quote.php（参看脚本11-1），在文本编辑器或IDE中打开它。

(2) 修改file_put_contents()行，代码如下（参看脚本11-2）：

```
file_put_contents($file, $_POST['quote'] . PHP_EOL, FILE_APPEND | LOCK_EX);
```

脚本11-2　add_quote.php脚本的修改版锁定了数据文件，因此更加安全可靠

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Add A Quotation</title>
7   </head>
8   <body>
9   <?php // 脚本11-1 - add_quote.php
10  /* 脚本显示并处理HTML表单。脚本从文本框中获取文本并将其存储到文本文件中。 */
11
12  // 标识要使用的文件名：
13  $file = '../quotes.txt';
14
15  // 检查表单是否提交：
16  if ($_SERVER['REQUEST_METHOD'] == 'POST') { // 处理表单。
17
18     if ( !empty($_POST['quote']) && ($_POST['quote'] != 'Enter your quotation
       here.') ) { // 检查输入文本。
19
20       if (is_writable($file)) { // 确认文件可以写入数据。
21
22          file_put_contents($file, $_POST['quote'] . PHP_EOL, FILE_APPEND | LOCK_EX);
            // 写入数据。
23
24          // 打印一条消息：
25          print '<p>Your quotation has been stored.</p>';
26
27       } else { // 无法打开文件。
28          print '<p style="color: red;">Your quotation could not be stored due to a
            system error.</p>';
29       }
30
31    } else { // 没有输入文本。
32      print '<p style="color: red;">Please enter a quotation!</p>';
```

```
33      }
34
35  } // 结束提交条件语句。
36
37  // 完成PHP节并显示表单:
38  ?>
39
40  <form action="add_quote.php" method="post">
41      <textarea name="quote" rows="5" cols="30">Enter your quotation here.</textarea><br/>
42      <input type="submit" name="submit" value="Add This Quote!" />
43  </form>
44
45  </body>
46  </html>
```

该命令给文件添加了一个独享锁，因此其他脚本不能同时对其进行写入。

同样，如果不能使用`file_put_contents()`和`LOCK_EX`，那就需要应用`flock()`函数：

```
flock($fp, LOCK_EX);
fwrite($fp, $_POST['quote'] . PHP_EOL);
flock($fp, LOCK_UN);
fclose($fp);
```

(3) 保存文件，将其放置到支持PHP的服务器上的适当目录中，并再次在Web浏览器中测试（参见图11-11和图11-12）。

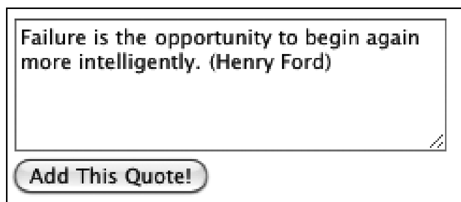


图11-11　再次使用该表单

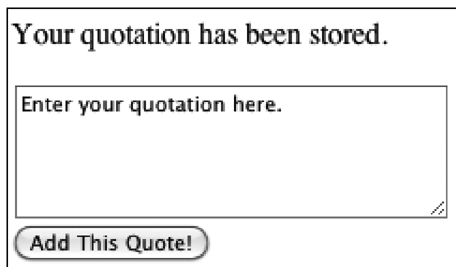


图11-12　存储文本时仍不会出现问题

✓提示

- 如果使用了`flock()`函数，当调用`fclose()`函数时，文件会自动解锁，但最好还是在文件写入完成后手动加上解锁语句。
- 从技术上讲，如果用追加模式打开了文件（如这个示例所示），即便不锁定文件，当多个脚本同时对文件进行写入时也不会产生什么问题。但是，这样做安全性会更高些。
- 要使用锁，则每个脚本在写入文件时都要使用锁。

11.4　读取文件

读取文件的方法很多，应该根据需要选择文件读取方式。使用`file_get_contents()`会将

文件中的所有内容按照一个字符串来读取：

```
$data = file_get_contents($file);
```

如果文件每一行都有一些数据，就像例子中的quotes.txt，最好使用file()函数：

```
$data = file($file);
```

file()函数是PHP中非常有价值的一个内置工具。与file_get_contents()不同，它读取文件中的所有内容并将这些信息放置在一个数组中。数组的每个元素都包含了文件中的一行，这些行是用换行符（\n或\r\n）分隔的。

如果$file指向的文件包含两行信息，每行都是以换行符结束的，则对应的数组会包含两个元素。第一个元素等价于$file中的第一行，而第二个元素等价于其中的第二行。如果数据被存储到数组中，就可以很方便地操作或打印它了，正如在第7章中学到的那样。

下面将使用这些知识创建一个脚本，随机显示一条之前存储的文本。

⇛ 读取一个文件

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为view_quote.php（参看脚本11-3）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>View A Quotation</title>
</head>
<body>
<h1>Random Quotation</h1>
```

脚本11-3　view_quote.php文件检索文本文件中所有的文本，并随机显示一条

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>View A Quotation</title>
7   </head>
8   <body>
9   <h1>Random Quotation</h1>
10  <?php //脚本11-3 - view_quote.php
11  /* 脚本显示并处理HTML表单。脚本从文本框中获取文本并将其存储到文本文件中。*/
12
13  // 将文件的内容写入数组：
14  $data = file('../quotes.txt');
15
16  // 获取数组中的元素个数：
17  $n = count($data);
18
19  // 获取一个随机元素：
20  $rand = rand(0, ($n - 1));
```

```
21
22  // 打印文本：
23  print '<p>' . trim($data[$rand]) . '</p>';
24
25  ?>
26  </body>
27  </html>
```

(2) 打开PHP代码段：

```
<?php // 脚本11-3 - view_quote.php
```

(3) 读取文件内容并将它们保存在一个数组中：

```
$data = file('../quotes.txt');
```

该函数将文件数据读取到一个名为$data的数组中。$data的每个元素都是一个字符串，对应于一条之前提交的文本。

如果quotes.txt文件并不位于该脚本的父目录中，请适当地修改这里的引用。

(4) 基于$data中元素的数量获取一个随机数：

```
$n = count($data);
$rand = rand(0, ($n - 1));
```

第一步获取了$data数组中元素的数量（也就是说，有多少条文本）。然后使用rand()函数选择一个随机数，需要使用一些逻辑。

如果$data有10个元素，它们的索引应该是0～9，所以在这里使用这样一个区间作为rand()的参数。在文本文件行数不同的情况下，应该以0和$data中元素的数量减1作为区间。

(5) 打印文本：

```
print '<p>' . trim($data[$rand]) . '</p>';
```

一条简单的print语句连接了要打印的随机文本。要检索该文本，需要使用前面生成的$rand数值作为索引来引用$data数组。在这之后截去了该本文末尾的换行符。

(6) 完成PHP代码和HTML页面：

```
?>
</body>
</html>
```

(7) 将文件保存为view_quote.php，放置在Web服务器上（和add_quote.php放在同一个目录中），并在Web浏览器中测试（参见图11-13）。

Random Quotation

Nurture your mind with great thoughts, for you will never go any higher than you think. (Benjamin Disraeli)

图11-13　每次查看页面时都会随机显示一条文本

(8) 在Web浏览器中重新加载页面，浏览另外一条随机观点（参见图11-14）。

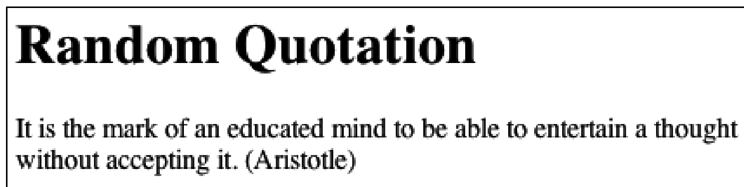


图11-14　再次访问view_quote.php脚本会显示来自文本文件中的另一条文本

✓提示

- 如果想要更谨慎一些，可以在调用file()函数之前使用is_readable()函数来测试一下PHP是否可以读取这个文件。
- readfile()函数用于读取一个文件并将内容发送到Web浏览器窗口。
- 在本章稍后的部分中，还将学到fgets()和fgetcsv()这两种用以读取文件的更为复杂的方式。

11.5　处理文件上传

正如本书已经演示过的那样，用PHP处理HTML表单是一个非常简单的过程。无论提交的数据是什么，PHP都能简单并且直观地处理。当用户通过HTML表单上传文件时也是如此。

为了让用户能够上传文件，必须对标准的HTML表单作出3处更改。首先，初始form标签必须包含代码enctype="multipart/form-data"，让浏览器知道表单数据将具备不同的类型：

```
<form action="script.php" enctype="multipart/form-data" method="post">
```

表单必须使用POST方法。

其次，必须添加一个特殊的隐藏输入框：

```
<input type="hidden" name="MAX_FILE_SIZE" value="30000" />
```

它用来建议浏览器最大能够上传多大的文件（以字节计）。

最后，使用file元素创建所需的表单字段（参见图11-15和图11-16）。

```
<input type="file" name="picture" />
```

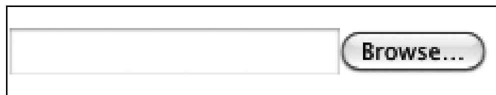


图11-15　Firefox这样解释file类型的输入框

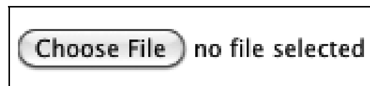


图11-16　Safari Web浏览器这样解释文件输入框（选择文件的按钮在前面）

file类型的表单输入框允许用户从他们自己的计算机上选择一个文件，然后提交，从而上传到服务器。上传完毕后，就能用PHP处理该文件了。

在PHP脚本中，可以引用$_FILES变量（文件领域中的$_POST等价物）来引用上传的文件。$_FILES数组包含5个元素：

- name，文件在用户计算机上使用的名字；
- type，文件的MIME类型（例如，image/jpg）；
- size，文件的大小（以字节计）；
- tmp_name，文件存放在服务器上的临时文件名；
- error，如果发生了错误则保存错误代码（参见表11-3）。

表11-3　**$_FILES**错误代码

代　码	意　义
0	没有错误发生
1	文件大小超过了php.ini中的upload_max_filesize设置
2	文件超过了HTML表单中的MAX_FILE_SIZE设置
3	文件只有部分被上传
4	未上传文件
6	临时目录不存在
7	写入磁盘出错
8	上传文件的扩展名被阻止

注意，错误代码中没有代码5，是不是很奇怪？

当文件上传后，服务器会将其放置在一个临时目录中。之后可以使用move_uploaded_file()函数将文件存放在其最终目标位置：

```
move_uploaded_file($_FILES['picture']['tmp_name'], '/path/to/dest/filename');
```

第一个参数是服务器中文件的临时名。第二个参数是移动目标的完整路径和文件名。

为了使PHP能够完成这些步骤，必须在php.ini文件中进行一些配置（参见框注“配置PHP以上传文件”），并且Web服务器必须对临时目录和最终的目标目录具有写权限。（默认情况下，PHP对临时目录有写访问权限。）

下面编写一个基本的脚本，完成文件上传并将其存储到服务器上。和add_quote.php脚本类似，该示例同时创建并处理HTML表单（参见图11-17），所有这些都在一个页面中完成。不过，首先需要在目标位置创建一个可写的目录。

图11-17　该HTML表单允许用户从他们的计算机上选择一个文件上传到服务器上

⇛ 要创建可写目录

(1) 创建一个新的文件夹，名为uploads，放置在Web根目录之外（参见图11-18）。

(2) 使用本章11.1节中列出的步骤，将该目录的权限设置为所有用户可读、可写和可搜索（在Unix中是0777）。

同样，如果运行的是Windows，则可能什么也不用做（直接尝试下面的脚本看看是否可行）。如果运行的是其他操作系统，请查看建议列表，找到最适合自己环境的方式。

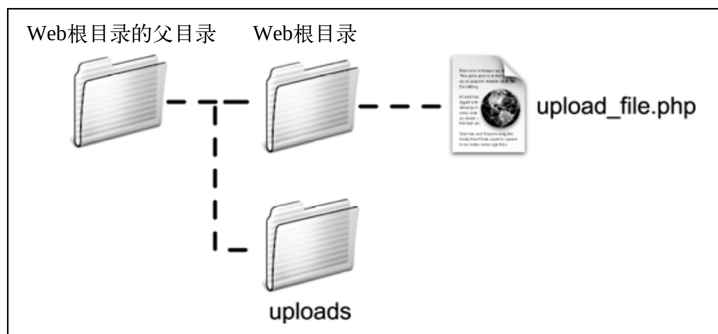


图11-18　对于这个示例，必须具有一个可写的uploads目录。在这里，将其放置到了和Web根目录一样的文件夹中。因此uploads位于upload_file.php脚本所在目录的上一级目录，并且不能通过Internet访问到

⇛ 使用PHP上传文件

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为upload_file.php（参看脚本11-4）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Upload a File</title>
</head>
<body>
```

脚本11-4　该脚本处理了文件上传，首先定义了适当的HTML表单，然后调用了move_uploaded_file()将文件移动到期望的位置

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Upload a File</title>
7   </head>
8   <body>
```

```
9   <?php // 脚本11-4 - upload_file.php
10  /* 脚本显示并处理HTML表单。脚本从文本框中获取文本并将其存储到文本文件中。*/
11
12  if ($_SERVER['REQUEST_METHOD'] == 'POST') { // 处理表单。
13
14    // 尝试移动上传文件:
15    if (move_uploaded_file ($_FILES['the_file']['tmp_name'], "../uploads/
      {$_FILES['the_file']['name']}")) {
16
17      print '<p>Your file has been uploaded.</p>';
18
19    } else { // 移动出错!
20
21      print '<p style="color: red;">Your file could not be uploaded because: ';
22
23      // 根据错误类型打印对应的错误信息:
24      switch ($_FILES['the_file']['error']) {
25         case 1:
26            print 'The file exceeds the upload_max_filesize setting in php.ini';
27            break;
28         case 2:
29            print 'The file exceeds the MAX_FILE_SIZE setting in the HTML form';
30            break;
31         case 3:
32            print 'The file was only partially uploaded';
33            break;
34         case 4:
35            print 'No file was uploaded';
36            break;
37         case 6:
38            print 'The temporary folder does not exist.';
39            break;
40         default:
41            print 'Something unforeseen happened.';
42            break;
43      }
44
45      print '.</p>'; // 段落结束。
46
47     } // 结束move_uploaded_file()条件语句。
48
49  } // 结束提交条件语句。
50
51  // 完成PHP节并显示表单:
52  ?>
53
54  <form action="upload_file.php" enctype="multipart/form-data" method="post">
55     <p>Upload a file using this form:</p>
56     <input type="hidden" name="MAX_FILE_SIZE" value="300000" />
57     <p><input type="file" name="the_file" /></p>
58     <p><input type="submit" name="submit" value="Upload This File" /></p>
59  </form>
60
61  </body>
```

```
62  </html>
```

(2) 创建PHP代码段：

```
<?php // 脚本11-4 - upload_file.php
```

(3) 检查表单是否已经被提交：

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

同样，该脚本同时显示并处理HTML表单。如果表单已经被提交，则应该处理上传的文件。

(4) 尝试将上传的文件移动到最终的目标位置：

```
if (move_uploaded_file ($_FILES['the_file']['tmp_name'],
→"../uploads/{$_FILES['the_file']['name']}")) {
```

move_uploaded_file()函数尝试将上传的文件（通过$_FILES['thefile']['tmp_name']指定）移动到新位置（../uploads/{$_FILES['thefile']['name']}）。其位置是uploads目录，位于当前脚本所在位置的上一级目录中。文件的名字将会和用户计算机上的名字相一致。

将该函数放在if语句的条件中，可以简单地根据是否移动成功做出响应。

(5) 打印消息，标明操作是否成功：

```
print '<p>Your file has been uploaded.</p>';
} else { // 移动出错！
  print '<p style="color: red;">Your file could not be uploaded because: ';
```

如果移动成功（参见图11-19），则执行第一个print语句。如果没有成功，则应用else子句，这时会开始打印错误消息。第6步更明确地给出了错误消息。

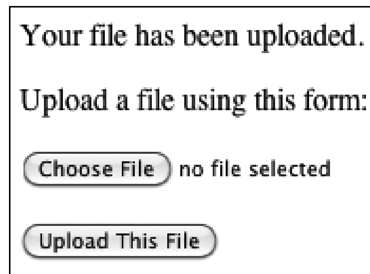


图11-19　如果上传和移动文件成功，则打印一条消息并再次显示表单

(6) 如果没有移动成功，则打印错误消息：

```
switch ($_FILES['the_file']['error']) {
  case 1:
    print 'The file exceeds the upload_max_filesize setting in php.ini';
    break;
  case 2:
    print 'The file exceeds the MAX_FILE_SIZE setting in the HTML form';
    break;
  case 3:
```

```
        print 'The file was only partially uploaded';
        break;
      case 4:
        print 'No file was uploaded';
        break;
      case 6:
        print 'The temporary folder does not exist.';
        break;
      default:
        print 'Something unforeseen happened.';
        break;
    }
```

如果没有移动成功，则$_FILES['the_file']['error']变量中将包含一个表示对应的错误消息的数字。通过使用switch条件语句，该PHP脚本可以打印出对应的错误消息（参见图11-20）。

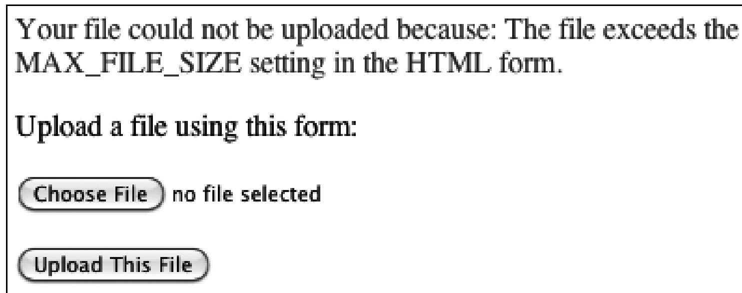


图11-20　如果发生错误，该脚本会指出错误原因

通常不应该将这样的代码放置到公开站点中（其中的信息太多了），但这对于调试代码中的问题来说，则大有益处。

(7) 完成错误消息，关闭两个条件语句：

```
        print '.</p>'; // Complete the paragraph.
      } // 结束move_uploaded_file()条件语句。
    } // 结束提交条件语句。
```

(8) 退出PHP代码节并创建HTML表单：

```
?>
<form action="upload_file.php" enctype="multipart/form-data" method="post">
  <p>Upload a file using this form:</p>
  <input type="hidden" name="MAX_FILE_SIZE" value="300000" />
  <p><input type="file" name="the_file" /></p>
  <p><input type="submit" name="submit" value="Upload This File" /></p>
</form>
```

该HTML表单非常简单，只包含两个可见的元素：一个file类型的输入框，和一个提交按钮。与本书中其他HTML表单的不同之处在于，它使用了enctype属性和一个MAX_FILE_SIZE隐藏输入框。

在为file输入框设置name属性的时候要很小心，因为这个值必须与$_FILES变量中使用的索引精确匹配。在这里使用普通的the_file。

(9) 完成HTML页面：

```
</body>
</html>
```

(10) 将页面保存为upload_file.php，放置在支持PHP的服务器上相对于uploads目录的适当目录中（参见图11-18），并在Web浏览器中进行测试（参见图11-21）。

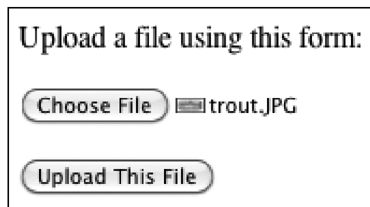


图11-21　在计算机上选择一个文件进行上传

由于MAX_FILE_SIZE的限制，只有小于300 KB的文件是允许的。

(11) 检查uploads目录，确认文件已被放置到这里。

✓提示

- 如果无法移动文件并显示权限被拒绝错误，请检查uploads目录的权限。然后检查脚本中使用的目录路径是否正确，并确定没有拼写错误。
- 你也许会发现，通过Web浏览器上传的文件（从权限上看）是属于Web服务器应用程序的。
- 从安全的角度来说，最好重命名上传的文件。要完成这一任务，需要设计一个系统，能够生成一个新的唯一的文件名，并在一个文本文件或数据库中同时保存原始文件名和新文件名。
- 一个脚本可以同时处理多个文件上传，只要它们的名字不同即可。在这种情况下，需要一个MAX_FILE_SIZE隐藏输入框来处理。在PHP脚本中，需要将move_uploaded_file()函数应用于$_FILES['filename1']、$_FILES['filename2']，依此类推。
- 通过在PHP脚本中引用适当的索引（例如，$_FILES['thefile'] ['size']），可以限制文件为确定的大小或类型（在文件上传之后进行）。
- 使用unlink()可以删除文件而不进行移动或复制。
- 使用copy()函数可以在服务器上创建文件的副本。

配置PHP以上传文件

为了能够上传文件，必须设置php.ini配置文件中的很多选项。对于你的配置来说，这些选项可能已经启用或尚未启用，因此需要查看php.ini文件或运行phpinfo()脚本来检查其配置情况。

首先，file_uploads必须启用。其次，必须将upload_tmp_dir值设置为服务器上PHP能够放置文件的目录（换句话说，该目录必须存在并且能够被Web服务器修改）。如果这个设置没有值，可能也没问题（这意味着将会专门创建一个隐藏目录）。

upload_max_filesize和post_max_size设置指定了可以传送多大的文件。表单中的

MAX_FILE_SIZE隐藏输入框只是对Web浏览器的建议，而这两个设置则控制着文件是否能够上传。

最后，如果的确要上传大文件（几MB甚至更大），可能需要增加memory_limit和max_execution_time设置的值，给PHP足够的空间和资源来完成它的工作。

11.6　导航目录

前面的PHP脚本用于处理文件，但使用PHP也可以针对目录做很多事情。在这个示例中，我们将编写一个脚本列出目录的内容，但首先需要理解所需使用的各种函数的用途和语法。

要查找一个目录中的所有内容，最简单的选择是使用scandir()函数：

```
$stuff = scandir($dir);
```

该函数是PHP 5中添加的，用于返回一个数组，其中包含了从给定目录中找到的所有项——目录和文件。就这个与文件有关的函数来说，$dir的值可以是绝对路径也可以是相对路径。

如果使用的是旧版本的PHP，就需要使用opendir()、readdir()和closedir()代替。查看PHP手册，了解readdir()函数的全部语法和用法。

下一个例子会使用scandir()函数，但我们先来看另外几个函数。在这个例子里会用到filesize()函数，它用于检查文件的大小（以字节计算）。可以将这个值赋给一个变量或打印出来：

```
$size = filesize($file);
```

类似地，filemtime()函数用于检索文件的修改时间。它会返回一个时间戳，可以使用date()函数格式化这个时间戳。

最后，PHP包含了很多用于获取文件属性的函数。本章已经提到过is_writable()和is_readable()，此外还有is_dir()和is_file()。如果一个项是文件或是目录，则对应的函数就会返回TRUE。

下面会将所有这些功能用在同一个页面中，该页面将形成一个基于Web的控制面板，用于查看目录的内容（参见图11-22）。

⇒ 创建目录控制面板

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为list_dir.php（参看脚本11-5）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Directory Contents</title>
</head>
<body>
```

Directories

- ch1
- ch2
- ch3
- templates

Files

Name	Size	Last Modified
add_quote.php	1520 bytes	December 29, 2010
list_dir.php	1562 bytes	December 28, 2010
login.php	1690 bytes	December 28, 2010
register.php	2339 bytes	December 28, 2010
upload_file.php	1844 bytes	December 29, 2010
view_quote.php	773 bytes	December 29, 2010

图11-22　list_dir.php脚本展示了一个目录中的内容。上面列出了子文件夹，下面的表格列出了文件

脚本11-5　该脚本显示了一个目录的内容。首先列出了子文件夹，然后在一个表格中列出了文件（包括文件大小和修改时间）

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Directory Contents</title>
7   </head>
8   <body>
9   <?php // 脚本11-5 - list_dir.php
10  /* 脚本显示一个目录中的文件和子目录。*/
11
12  // 设置时区：
13  date_default_timezone_set('America/New_York');
14
15  // 设置目录名并扫描其内容：
16  $search_dir = '.';
17  $contents = scandir($search_dir);
18
19  // 列出目录中的子目录：
20  // 以原点开始，后接目录名：
21  print '<h2>Directories</h2>
22  <ul>';
23  foreach ($contents as $item) {
24    if ( (is_dir($search_dir . '/' . $item)) AND (substr($item, 0, 1) != '.') ) {
25       print "<li>$item</li>\n";
```

```
26        }
27    }
28
29    print '</ul>'; // 关闭列表。
30
31    // 创建一个表头:
32    print '<hr /><h2>Files</h2>
33    <table cellpadding="2" cellspacing="2" align="left">
34    <tr>
35    <td>Name</td>
36    <td>Size</td>
37    <td>Last Modified</td>
38    </tr>';
39
40    // 列出目录下的所有文件:
41    foreach ($contents as $item) {
42       if ( (is_file($search_dir . '/' . $item)) AND (substr($item, 0, 1) != '.') ) {
43
44          // 获取文件大小:
45          $fs = filesize($search_dir . '/' . $item);
46
47          // 获取文件的修改时间:
48          $lm = date('F j, Y', filemtime($search_dir . '/' . $item));
49
50          // 打印信息:
51          print "<tr>
52          <td>$item</td>
53          <td>$fs bytes</td>
54          <td>$lm</td>
55          </tr>\n";
56
57       } // 结束条件语句。
58
59    } // 结束FOREACH。
60
61    print '</table>'; // 关闭HTML表格。
62
63    ?>
64    </body>
65    </html>
```

(2) 打开PHP代码节并设置时区：

```
<?php // 脚本11-5 - list_dir.php
date_default_timezone_set('America/New_York');
```

因为该脚本要使用`date()`函数，所以需要建立一次时区。请参见第8章以查找更多信息，并能从这一章中找到参考PHP手册中的哪一部分可以找到所需的时区。

(3) 指定要打开的目录，并扫描其内容：

```
$search_dir = '.';
$contents = scandir($search_dir);
```

通过在PHP脚本的顶部将这些值建立为变量，可以根据需要很方便地查找和修改这些值。这

里使用圆点来引用当前目录。也可以使用绝对路径（/Users/larry/Documents或C:\\myfiles\\directory）或相对路径（../myfiles）来引用其他目录，只要PHP有权限读这个目录即可。

第二行扫描了目录的内容，并将其作为一个数组赋给变量$contents。

(4) 列出该目录的子目录：

```
print '<h2>Directories</h2>
<ul>';
foreach ($contents as $item) {
  if ( (is_dir($search_dir . '/' . $item)) AND (substr($item, 0, 1) != '.') ) {
    print "<li>$item</li>\n";
  }
}
print '</ul>';
```

foreach循环访问了数组中的每一项，每次都将其中一项赋给$item变量。因为首先希望列出每个子目录，所以使用is_dir()函数来确认项的类型。然后检查它并不是在当前目录（在Unix系统上使用一个单独的圆点表示）。如果该条件的结果是TRUE，则打印出项的名字，将其放置在列表项标签内，然后跟一个换行符（使生成的HTML源代码更整洁）。

这样在每一项前加上$search_dir和反斜杠，is_dir()函数就也可以用于检查其他目录中的项了。如果代码仅引用$item变量，而没有添加目录路径。那代码仅在当前目录下生效。

(5) 新建一个标题，并为了罗列文件打开一个表格：

```
print '<hr /><h2>Files</h2>
<table cellpadding="2" cellspacing="2" align="left">
<tr>
<td>Name</td>
<td>Size</td>
<td>Last Modified</td>
</tr>';
```

该脚本还要显示文件的大小和修改时间。为了使这些信息看起来更漂亮，这里将结果放置在了HTML表格中。

(6) 开始循环遍历该目录中的文件：

```
foreach ($contents as $item) {
  if ( (is_file($search_dir . '/' . $item)) AND (substr($item, 0, 1) != '.') ) {
```

这里使用另外一个foreach循环再次遍历了目录的内容。这一次只需要文件项（但不包括以圆点开始的隐藏文件）。

同样，在每个项目之前会添加$search_dir和一个反斜杠。

(7) 计算文件的大小和修改时间，并打印这些信息：

```
$fs = filesize($search_dir . '/' . $item);
$lm = date('F j, Y', filemtime ($search_dir . '/' . $item));
print "<tr>
<td>$item</td>
<td>$fs bytes</td>
```

```
<td>$lm</td>
</tr>\n";
```

第一行调用`filesize()`函数获取了文件的大小（以字节计算）。第二行调用`filemtime()`函数，它会返回反映了文件修改时间的时间戳。紧接着这个值被传递给了`date()`函数，通过进行适当的格式化，返回类似November 24, 2011这样的字符串。最后，这两项以及文件的名字会打印在表格中的适当列中。

(8) 完成条件语句和循环：

```
  }
}
```

(9) 关闭表格：

```
print '</table>';
```

(10) 完成PHP代码和HTML页面：

```
?>
</body>
</html>
```

(11) 将文件保存为`list_dir.php`，将其放置在启用了PHP的服务器上的适当目录中，并在Web浏览器中测试（参见图11-22）。

(12) 你可以更改`$search_dir`的值，并在Web浏览器中重新测试脚本（参见图11-23）。

Directories

- ch11
- css
- temp
- templates

Files

Name	Size	Last Modified
add_quote.php	1520 bytes	December 29, 2010
books.php	514 bytes	December 13, 2010
calculator.html	1094 bytes	October 16, 2008
calculator.php	1709 bytes	October 27, 2008
customize.php	1694 bytes	December 20, 2010
event.html	1043 bytes	December 7, 2010

图11-23　服务器上另一个文件夹的目录列表

✓提示

❑ 注意，在Windows服务器上，需要使用双反斜线来创建绝对路径名，因为Windows路径名中的单反斜线会导致字符被转义。因此，必须对单反斜线进行转义。

- 使用glob()函数可以搜索名字与模式相匹配的文件目录（如*.jpg或filename*.doc）。
- 可能会用到的其他文件函数包括fileperms()，它返回文件的权限；fileatime()用以返回文件的最后访问时间；以及fileowner()，它返回拥有该文件的用户名称。
- basename()或dirname()函数可以用来返回完整路径字符串中的文件名或目录名。
- finfo_file()函数用于检测文件的MIME类型。

11.7 创建目录

理解如何在服务器上读取和写入文件只是数据存储过程中的一个部分。为了实现我们的目标，可能还需要使用目录。

PHP中用于创建目录的命令是mkdir()：

```
mkdir('directory_name', permissions);
```

*directory_name*是要创建的目录的名字。这个值可以是以当前目录（即脚本所在的目录）为起点的相对路径，也可以是完整路径：

```
mkdir('C:\\inetpub\\users\\george');
```

在Windows服务器上，权限会被忽略，因此该参数不是必需的（如前面这个示例）。在其他服务器上，权限的默认值是0777（关于这个数值的意义，请参见本章中11.1节）。

记住这一点之后，我们来建立一个脚本，为每一个新注册的用户创建一个目录。该脚本还会将用户的用户名和密码记录在一个文本文件中，因此在登录时可以对用户进行验证。首先需要创建父目录（必须是可写的，这样PHP才能在其中创建子目录）和users.txt数据文件。

⇛ 创建目录和数据文件

(1) 新建一个名为users的文件夹，放在Web根目录之外。

可以将其放置在和之前创建的uploads文件夹相同的位置（参见图11-18）。

(2) 使用本章第一节给出的步骤，将该文件夹的权限设置为任何用户可读、可写和可搜索（在Unix中是0777）。

如果正在运行Windows，则很可能不需要这一步。

(3) 在文本编辑器中创建一个新的空白文档。

(4) 将该文件保存为users.txt并放置在users目录中。

(5) 再次使用本章前面列出的步骤，将users.txt的权限设置为所有用户可写、可读（在Unix中是0666）。

如果所用的PHP服务器上运行的是Windows，则这一步也不是必需的。

✓提示

- 如果创建了一个PHP可以写入的目录，PHP会在该目录中自动创建一个可写的users.txt文件。但最好还是不要依赖这种自动的行为。

⇒ 创建注册脚本

(1) 在文本编辑器或IDE中打开一个新的PHP文档，命名为register.php（参看脚本11-6）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>Register</title>
    <style type="text/css" media="screen">
      .error { color: red; }
    </style>
</head>
<body>
<h1>Register</h1>
```

脚本11-6　register.php脚本用于两个目的：将用户信息记录到一个文本文件中，以及为用户的内容创建一个新目录

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Register</title>
7      <style type="text/css" media="screen">
8        .error { color: red; }
9      </style>
10  </head>
11  <body>
12  <h1>Register</h1>
13  <?php // 脚本11-6 - register.php
14  /* 脚本将用户信息记录到一个文本文件中，并为用户创建一个新目录。*/
15
16  // 标识要使用的目录和文件：
17  $dir = '../users/';
18  $file = $dir . 'users.txt';
19
20  if ($_SERVER['REQUEST_METHOD'] == 'POST') { // 处理表单。
21
22     $problem = FALSE; // 目前一切正常。
23
24     // 检查注册信息……
25     if (empty($_POST['username'])) {
26        $problem = TRUE;
27        print '<p class="error">Please enter a username!</p>';
28     }
29
30     if (empty($_POST['password1'])) {
31        $problem = TRUE;
32        print '<p class="error">Please enter a password!</p>';
33     }
```

```
34
35     if ($_POST['password1'] != $_POST['password2']) {
36        $problem = TRUE;
37        print '<p class="error">Your password did not match your confirmed password!</p>';
38     }
39
40     if (!$problem) { // 如果一切正常……
41
42        if (is_writable($file)) { // 打开文件。
43
44           // 创建要写入的数据：
45           $subdir = time() . rand(0, 4596);
46           $data = $_POST['username'] . "\t" . md5(trim($_POST['password1'])) .
             "\t" . $subdir . PHP_EOL;
47
48           // 写入数据：
49           file_put_contents($file, $data, FILE_APPEND | LOCK_EX);
50
51           // 创建目录：
52           mkdir ($dir . $subdir);
53
54           // 打印一条消息：
55           print '<p>You are now registered!</p>';
56
57        } else { // 无法写入文件。
58           print '<p class="error">You could not be registered due to a system error.</p>';
59        }
60
61     } else { // 注册信息不完整。
62        print '<p class="error">Please go back and try again!</p>';
63     }
64
65  } else { // 显示表单。
66
67  // 结束PHP节并显示表单：
68  ?>
69
70  <form action="register.php" method="post">
71     <p>Username: <input type="text" name="username" size="20" /></p>
72     <p>Password: <input type="password" name="password1" size="20" /></p>
73     <p>Confirm Password: <input type="password" name="password2" size="20" /></p>
74     <input type="submit" name="submit" value="Register" />
75  </form>
76
77  <?php } // 结束提交条件语句?>
78  </body>
79  </html>
```

在页面的head中，定义了一个CSS类，用于格式化错误消息。

(2) 打开PHP代码：

```
<?php // 脚本11-6 - register.php
$dir = '../users/';
$file = $dir . 'users.txt';
```

这两个变量表示这个示例正在使用的目录和文件。文件位于目录中，所以变量的值以目录开始。根据你的实际情况修改的$dir的值。

(3) 检查表单是否已经被提交：

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

该页面再次同时显示和处理HTML表单。这是通过一个检查脚本的请求方式的条件语句来完成的。

(4) 验证注册信息：

```
$problem = FALSE;
if (empty($_POST['username'])) {
  $problem = TRUE;
  print '<p class="error">Please enter a username!</p>';
}
if (empty($_POST['password1'])) {
  $problem = TRUE;
  print '<p class="error">Please enter a password!</p>';
}
if ($_POST['password1'] != $_POST ['password2']) {
  $problem = TRUE;
  print '<p class="error">Your password did not match your confirmed password!</p>';
}
```

与本书之前开发的注册表单相比，这个注册表单是一个更加简单的版本。这里检查了提交的用户名和密码，验证的过程和之前开发的一样。$problem变量用于指明是否出现了问题。

(5) 检查错误：

```
if (!$problem) {
```

在这里，$problem变量可以用以知道是否能够注册该用户。如果没有出现错误，则继续向下进行将是安全的。

(6) 检查users.txt是否可以写入数据：

```
if (is_writable($file)) {
```

和前面的例子一样，用条件语句检查文件是否可写，脚本可以根据情况报告文件是否可写。如果使用的版本不是PHP 5.1或以上版本，在条件语句中使用fopen()函数（参看11.2节）。

(7) 创建要向文件中写入的数据，然后将它写入：

```
$subdir = time() . rand(0, 4596);
$data = $_POST['username'] . "\t" . md5(trim($_POST['password1'])) . "\t" . $subdir .
→PHP_EOL;
file_put_contents($file, $data, FILE_APPEND | LOCK_EX);
```

目录的名字由一个基于用户注册时间的数字和一个随机值组成。这样的系统有助于保证目录的名字既是唯一的也是有效的。

这个脚本没有像之前那样存储一个单独的字符串，而是存储了3块信息：用户名、密码的加密版本（使用md5()函数，请参见第一个提示）和前面几行创建的目录名字。首先截取密码，去掉了无关的空格。

为了区分各块信息，在它们之间插入了一个制表符（通过代码中的\t生成）。换行符用于标志行的结尾仍旧使用PHP_EOL常量。

(8) 创建用户的目录，并打印一条消息：

```
mkdir ($dir . $subdir);
print '<p>You are now registered!</p>';
```

mkdir()函数在users目录中创建了子目录。目录的名字是前面生成的随机数。

(9) 完成条件语句：

```
  } else { // 无法写入文件。
    print '<p class="error">You could not be registered due to a system error.</p>';
  }
} else { // 注册信息不完整。
  print '<p class="error">Please go back and try again!</p>';
}
```

第一个else完成了判断是否能够打开users.txt进行写入的条件语句（参见图11-24）。第二个else完成了判断用户能否正确完成表单处理的条件语句（参见图11-25）。



图11-24　如果users.txt不可写，会出现的结果

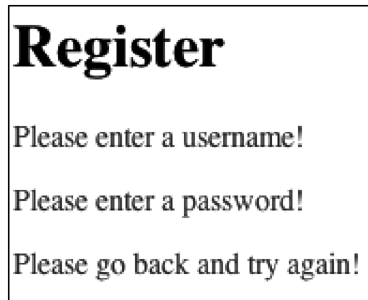


图11-25　脚本报告表单所有验证错误

(10) 向主条件语句添加else子句，并完成PHP节：

```
} else {
?>
```

该PHP脚本首先显示表单，然后处理它，这和本章之前的示例有所不同。之前的脚本在处理完表单后都会再次显示该表单，而这个脚本不会，它会在else语句中再创建一个表单。由于页面的其他部分只是HTML，因此这里暂时离开PHP片段去创建表单。

(11) 显示HTML表单：

```
<form action="register.php" method="post">
  <p>Username: <input type="text" name="username" size="20" /></p>
  <p>Password: <input type="password" name="password1" size="20" /></p>
  <p>Confirm Password: <input type="password" name="password2" size="20" /></p>
  <input type="submit" name="submit" value="Register" />
</form>
```

(12) 完成主条件语句：

```
<?php } // 结束提交条件语句。?>
```

最后的右花括号关闭了主提交条件语句。在使用它之前，必须首先在此打开PHP片段。

(13) 完成HTML页面：

```
</body>
</html>
```

(14) 将文件保存为register.php，放置到启用了PHP的服务器上的适当目录中，然后在Web浏览器中测试（参见图11-26和图11-27）。

Register

Username: larry

Password: ••••

Confirm Password: ••••

Register

图11-26　注册表单非常简单，但功能齐全



图11-27　注册成功后，用户将会看到该消息

(15) 如果需要的话，可以在文本编辑器中打开users.txt文件查看其内容（参看脚本11-7）。

脚本11-7　Users.txt列出了3列以tab划分的字段信息：用户名、加密的用户密码以及相关的目录名

```
1   larry    1a1dc91c907325c69271ddf0c944bc72    12936537501284
2   john     0cc175b9c0f1b6a831c399e269772661    1293653788455
3   paul     92eb5ffee6ae2fec3ad71c777531578f    12936537931717
4   george   4a8a08f09d37b73795649038408b5f33    1293653799360
5   ringo    8277e0910d750195b448797616e091ad    12936538042144
```

✓提示

- ❑ md5()函数创建了一个散列值，它是字符串经过数据计算后的一种表示。所以这个脚本并不真的存储密码，而是密码的一个表示（从理论上讲，不会有两个字符串具有相同的md5()值）。很快就会看到这个函数在登录脚本中的使用方式。
- ❑ 查看users目录中是否出现了新的子目录，通过这种方式也可以确认页面是否按照所期望的方式运行。
- ❑ 如果PHP拥有权限，则可以使用rmdir()函数来删除现有目录。

11.8　增量读取文件

在view_quote.php脚本（参看脚本11-3）中，我们使用file()函数将整个文件读取到了一个数组中。但是如果某一时刻只需要读取文件的一小部分呢？这时就需要使用fgets()函数了。

fgets()函数返回具有指定长度的字符串。通常将该函数放置在while循环中，并用feof()函数来确保没有到达文件的结尾。例如：

```
$fp = fopen($file, 'rb');
while (!feof($fp)) {
        $string = fgets($fp, 1024);
}
fclose ($fp);
```

在这个例子中，fgets()函数每次返回1023字节的数据（指定的长度1024再减1），直到行末尾或文件结尾。长度参数是可选的，但如果选用的话，它应该使用一个比文件中每个单行文本长度都要大的数值。如果只想读到行的末尾，省略长度参数即可：

```
$string = fgets($fp);
```

有的例子中会采用更具描述性（delineated）的格式来存储数据（通常使用逗号分隔，因此是CSV，这种以逗号分隔值的格式），这时可以使用fgetcsv()函数。它使用给定的分隔符切分字符串，并返回一个数组：

```
$array = fgetcsv($fp, length, delimiter);
$array = fgetcsv($fp, 1024);
```

前面的函数调用也是会返回1023字节的数据，但是它使用默认的分隔符（逗号）对字符串进行切分，该分隔符用作元素位置的指示器。该函数等价于将fgets()和explode()函数结合到一起使用。如果提供分隔符参数，就可以改变用于描述数据的分隔符。

最后，因为这些函数依赖于行尾指示符，所以最好采用额外的保护措施，即启用PHP的auto_detect_line_endings设置。可以使用ini_set()函数来完成：

```
ini_set('auto_detect_line_endings', 1);
```

作为示例，我们下面使用前面示例创建的users.txt文件来创建一个登录脚本。它会持续地读取文件，直到找到一组匹配的用户名和密码。

⇛ 增量地读取文件

(1) 在文本编辑器或IDE中打开一个新的PHP文档，命名为login.php（参看脚本11-8）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Login</title>
</head>
<body>
<h1>Login</h1>
```


脚本11-8　login.php脚本使用存储在users.txt（创建于脚本11-6）中的信息来验证用户登录

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Login</title>
7   </head>
8   <body>
9   <h1>Login</h1>
10  <?php // 脚本11-8 - login.php
11  /* 脚本使用存储在文本文件中的信息，验证用户登录。*/
12
13  // 标识要使用的文件：
14  $file =  '../users/users.txt';
15
16  if ($_SERVER['REQUEST_METHOD'] == 'POST') { // 处理表单。
17
18      $loggedin = FALSE; // Not currently logged in.
19
20      // 启用auto_detect_line_settings:
21      ini_set('auto_detect_line_endings', 1);
22
23     // 打开文件：
24     $fp = fopen($file, 'rb');
25
26     // 循环遍历文件：
27     while ( $line = fgetcsv($fp, 200, "\t") ) {
28
29        // 比较文件中的数据和已提交的数据：
30        if ( ($line[0] == $_POST['username']) AND ($line[1] == md5(trim($_POST
          ['password']))) ) {
31
32           $loggedin = TRUE; // 用户名与密码与文件中相匹配。
33
34           // 停止遍历文件：
35           break;
36
37         } // 结束条件语句。
38
39     } // 结束循环。
40
41     fclose($fp); // 关闭文件。
42
43     // 打印一条消息：
44     if ($loggedin) {
45         print '<p>You are now logged in.</p>';
46     } else {
47         print '<p style="color: red;">The username and password you entered do not
           match those on file.</p>';
48     }
49
50  } else { // 显示表单。
51
```

```
52  // 结束PHP节并显示表单：
53  ?>
54
55  <form action="login.php" method="post">
56      <p>Username: <input type="text" name="username" size="20" /></p>
57      <p>Password: <input type="password" name="password" size="20" /></p>
58      <input type="submit" name="submit" value="Login" />
59  </form>
60
61  <?php } // 结束提交条件语句。?>
62
63  </body>
64  </html>
```

(2) 创建PHP代码段并指明使用的文件：

```
<?php // 脚本11-8 - login.php
$file =  '../users/users.txt';
```

变量`$file`的值应该与`register.php`中的相同。

(3) 检查表单是否已经被提交：

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

(4) 创建一个临时变量，用作标志：

```
$loggedin = FALSE;
```

`$loggedin`变量用于指出用户是否输入了正确的用户名/密码组合。当该脚本首次启动时，假设他们还没有输入正确的值。

(5) 打开用于读取的文件：

```
ini_set('auto_detect_line_endings', 1);
$fp = fopen($file, 'rb');
```

和`file()`函数不同，`fgetcsv()`函数需要一个文件指针。因此，必须使用`fopen()`函数以适当的模式打开`users.txt`文件。在这里，使用的模式是`rb`，它表示文件需要以二进制安全模式打开，以便读取。

首先，也是为了安全起见，启用了PHP的`auto_detect_line_encodings`设置。

(6) 循环遍历文件中的每一行：

```
while ( $line = fgetcsv($fp, 200, "\t") ) {
```

该`while`循环的每一次迭代都会读取文件中的200个字节或1行——看哪一个先达到。读到的数据将被分割到一个数组中，使用制表符作为分割元素的分隔符。

由于`users.txt`文件采用的数据存储格式是username tab password tab directory newline，因此`$line`数组包含3个元素，索引分别是`0`（用户名）、`1`（密码）和`2`（目录）。

(7) 比较提交的值和检索到的值：

```
if ( ($line[0] == $_POST['username']) AND ($line[1] ==
→md5(trim($_POST['password']))) ) {
```

这个条件语句有两部分，分别检查了提交的用户名和存储的用户名（$line[0]）是否相等，以及提交的密码是否和存储的密码（$line[1]）相等。然而，由于存储的密码是使用md5()加密过的，因此需要首先对提交的值应用md5()然后再进行比较。

(8) 如果找到了匹配的值，将$loggedin设置为TRUE并退出while循环：

```
$loggedin = TRUE;
break;
```

如果条件值是TRUE，则提交的用户名和密码与文件中的值是匹配的。在这种情况下，$loggedin标志被设置为TRUE，而break语句用于退出while循环。这种系统的优点在于，只需要读取找到匹配之前的那一块文件内容。

(9) 关闭条件语句、while循环，并关闭文件：

```
  }
}
fclose ($fp);
```

(10) 向用户打印一条消息：

```
if ($loggedin) {
  print '<p>You are now logged in.</p>';
} else {
  print '<p style="color: red;"> The username and password you
  →entered do not match those on file.</p>';
}
```

该脚本使用$loggedin标志告诉用户是否“已登入”。在这个过程中还可以添加一些额外的功能，如将用户目录保存到会话当中，并将用户带到一个文件上传页面。

(11) 继续编写主提交条件，并退出PHP：

```
} else {
?>
```

(12) 创建HTML表单：

```
<form action="login.php" method="post">
  <p>Username: <input type="text" name="username" size="20" /></p>
  <p>Password: <input type="password" name="password" size="20" /></p>
  <input type="submit" name="submit" value="Login" />
</form>
```

(13) 返回PHP以完成主条件语句：

```
<?php } // 结束提交条件语句。?>
```

(14) 完成HTML页面：

```
</body>
</html>
```

(15) 将文件保存为login.php，放置在启用了PHP的服务器上的适当目录中，并在Web浏览器中测试（参见图11-28、图11-29和图11-30）。

图11-28 login表单接收用户名和密码

图11-29 如果提交的用户名和密码与之前记录的值匹配，则用户会看到这样的消息

图11-30 如果用户提交的用户名和密码与之前记录的值不匹配，则会出现这样的结果

✓提示

- 在PHP 4.2中，`fgets()`中`length`参数是可选的并且默认值是1KB（1024字节），而在PHP 4.3中，`length`参数的默认值是自动返回换行符之前的所有数据。
- 在PHP 5.3中，`fgetcsv()`函数接受另一个可选参数：用于转义问题字符的字符。默认的转义字符是反斜杠。
- 如果某一行为空，`fgetcsv()`返回一个数组，其中只包含一个单独的null值。

11.9 回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

11.9.1 回顾

- 你使用的是哪个版本的PHP？
- 要使服务器上的文件或文件夹可写，需要哪些步骤？
- 什么是Web根目录（概念）？你的网站的Web根目录是什么（是在你自己的计算机上还是在远程服务器上）？
- 向文件写入数据的两种方式各是什么？
- 如何向已有文件中追加数据（相对于替换已有数据而言）？
- 如何确保每个新数据都独占一行？
- 要让表单接受文件上传，在`form`开始标签中应加入什么属性？

- 在哪个变量中PHP脚本能够访问上传文件？哪个函数可以将文件移动到服务器的最终目标位置？
- fgetcsv()函数与file()或file_get_contents()有什么不同？

11.9.2　实践

查看PHP手册中其他与文件系统相关的函数（起始网址：www.php.net/manual/en/ref.filesystem.php）。

修改add_quote.php，先检查quotes.txt文件是否存在，再检查文件是否可写。

将add_quote.php页面中的文本区变为粘性文本区。

修改add_quote.php用两个文本框分别接受引用语和署名，并写入到文本文件中。接下来修改view_quote.php使其获取并显示这些数据。

修改view_quote.php以显示两个随机的引用语。

修改upload_file.php，使其确认uploads文件夹可写。

查看PHP手册中glob()函数的信息，了解它的功能和使用方法。

更新list_dir.php，显示目录中文件的其他信息。

- 在register.php中创建一个验证系统，确保用户名唯一。提示：在创建目录之前，使用PHP检查已有用户名列表，查找是否有完全相同的用户名。如果没有找到相同的用户名，创建这个新的用户名。如果这个用户名已经存在，让PHP提示错误信息，让用户重新创建用户名。
- 结合使用写入、读取文本文件以及session或cookie技术，创建一个真实的注册登录系统。

第 12 章

数据库介绍

本章内容

- ❑ SQL介绍
- ❑ 连接MySQL
- ❑ MySQL错误处理
- ❑ 创建和选择数据库
- ❑ 创建表
- ❑ 向数据库插入数据
- ❑ 安全查询数据
- ❑ 从数据库中检索数据
- ❑ 删除数据库中的数据
- ❑ 更新数据库中的数据
- ❑ 回顾和实践

如果没有数据库的存在，Internet根本就不会是今天这个样子。实际上，如果没有对多种数据库提供内置的支持，PHP根本就不会这么流行或这么有用。本章将使用MySQL作为示例DBMS。尽管MySQL（可以应用在多种平台上）也许并不像其他数据库服务器那样强大，但对于大多数需求来说，它拥有足够的速度和功能。此外，它对于很多用户来说是免费的，这使它成为Web开发中最常见的选择。

本章介绍了如何利用简单的数据库来创建博客（一种基于Web的日记）。这里学到的东西足够让你达到入门级别，但学习完本章后，你若还需要找到一些参考资料来学习更多的内容，可以先看看附录B。

12.1 SQL 介绍

数据库（database）是存储了信息的表的集合（表由列和行构成）。数据库使用SQL（结构化查询语言）来进行创建、更新和读取。令人惊讶的是，SQL中有些命令（表12-1列出了最重要的7个）既是福也是祸。

表12-1　常用SQL命令

命　令	功　能
ALTER	修改已有表
CREATE	创建数据库或表
DELETE	从表中删除数据
DROP	删除数据库或表
INSERT	向表中添加记录
SELECT	从表中检索数据
UPDATE	更新表中的记录

SQL语句写起来很像英语，这使得它非常友好，但要想使用如此有限的术语来创建更加复杂的SQL语句，还是要花一些心思的。本章会讲授如何定义所有的基本SQL语句（也称为查询）。

PHP新手可能会对PHP和HTML的关系（PHP可以用于生成HTML，但PHP代码从来不在Web浏览器中运行）产生混淆。现在又有了数据库，其间关系变得更为模糊。其过程其实非常简单：PHP仅用来将SQL语句发送到数据库应用程序，那里才是SQL语句真正执行的地方。创建表、插入记录、检索到的一些记录甚至是错误，这些执行的结果会从数据库返回到PHP脚本（参见图12-1）。

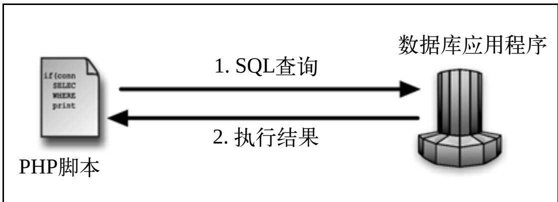


图12-1　PHP将SQL语句发送至MySQL。MySQL将执行语句并把结果返回给PHP脚本

考虑到这一点，PHP的`mysql_query()`函数将成为本章用到最多的工具。它可以将一条SQL命令发送到MySQL：

```
$result = mysql_query(SQL command, database connection);
```

本章开头就已经说明了这一点，因为SQL和MySQL的加入使得Web开发过程更为复杂了。当发生错误时（毋庸置疑，肯定会发生），需要知道如何最好地对其进行调试。

当PHP脚本没有与MySQL数据库按照预期方式交互时，首先要确定是不是由查询本身的问题引起的（参见图12-1中的“SQL查询”）；还是由查询的结果引起的（参见图12-1中的“执行结果”）。可以打印正在执行的查询来调试脚本，使用以下代码：

```
print $query;
```

`$query`代表一个完整的SQL命令，一般包含PHP变量的值，使用这个仅有一行的代码就可以显示正在运行的SQL语句。

接下来，使用其他应用程序执行上述语句打印的SQL语句。这里有两个常用的工具：

- ❑ MySQL客户端（参见图12-2），与MySQL交互的命令行工具；

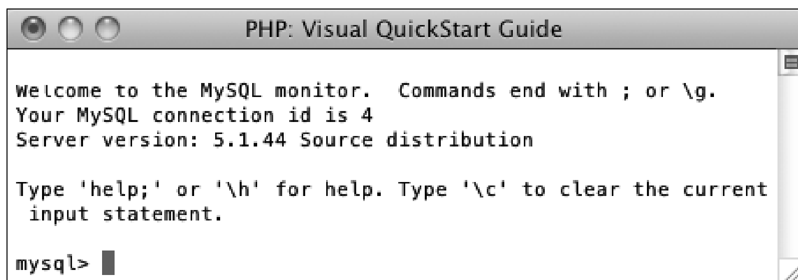


图12-2　MySQL客户端是MySQL数据库软件自带的，无需PHP脚本就能执行查询

- ❑ phpMyAdmin（参见图12-3），基于PHP的MySQL界面。

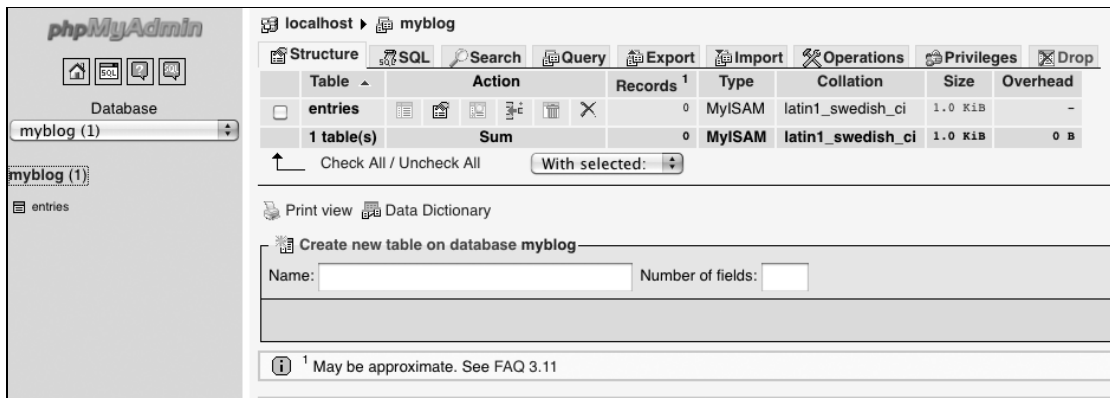


图12-3　phpMyAdmin可能是用PHP编写的最流行的软件了。它为MySQL数据库提供了基于Web的界面

托管公司应该会提供这两个（或其中之一）工具，或者你安装在本地机的软件中也会提供这些工具。关于使用这两个工具的说明，请参见附录A。

✓提示

- ❑ 从技术上说，DBMS或数据库应用程序是与对应的数据库进行接口的软件。不过，大多数人认为数据库和DBMS是同义词。
- ❑ 很多其他应用程序也能脱离MySQL客户端和phpMyAdmin与MySQL进行交互。有的是免费的，有的则需要花钱。使用Google快速搜索MySQL、admin和所用的操作系统，能够找到很多结果。

12.2　连接 MySQL

在第11章中操作文本文件时，已经了解到一些函数（如`fwrite()`和`fgets()`）在打开文件时必须首先创建一个文件指针（使用`fopen()`）。之后这个指针可以用作指向被打开文件的引用。

在使用数据库时，采用类似的过程。首先，需要建立一个到数据库服务器（这里是MySQL）的连接，之后该连接将可以用作其他命令的访问点。连接到数据库的语法是：

```
$dbc = mysql_connect(hostname, username, password);
```

数据库连接（$dbc）的建立至少需要3个参数：主机名（一般是localhost）、用户名和与用户名对应的密码。

如果使用的是主机公司的数据库，该公司很有可能会提供用户名和密码。如果是在自己的计算机上运行MySQL，请参见附录A学习如何创建用户。

一旦使用完数据库，就可以关闭数据库连接，就像关闭一个已打开的文件一样：

```
mysql_close($dbc);
```

PHP脚本会在脚本执行结束后自动关闭数据库连接，但最好是在不使用时显式地关闭数据库连接。

作为本章的第一个例子，下面将编写一个简单的脚本，尝试连接到MySQL。当这个连接可以工作之后，就能继续进行本章的其余部分了。

⇛ 连接到MySQL

(1) 在文本编辑器或IDE中启动一个新的PHP文档，命名为mysql_connect.php（参看脚本12-1）。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Connect to MySQL</title>
</head>
<body>
```

脚本12-1　能够连接到MySQL服务器是最重要的一步，这个脚本测试了这个过程

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Connect to MySQL</title>
7   </head>
8   <body>
9   <?php // 脚本12-1 - mysql_connect.php
10  /* 脚本连接到MySQL服务器。*/
11
12  // 尝试连接到MySQL并打印消息:
13  if ($dbc = mysql_connect('localhost', 'username', 'password')) {
14
15       print '<p>Successfully connected to MySQL!</p>';
16
17       mysql_close($dbc); // 关闭连接。
```

```
18
19  } else {
20
21      print '<p style="color: red;">Could not connect to MySQL.</p>';
22
23  }
24
25  ?>
26  </body>
27  </html>
```

(2) 打开PHP代码片段：

```
<?php // 脚本12-1 - mysql_connect.php
```

(3) 连接到MySQL，并报告结果：

```
if ($dbc = mysql_connect('localhost', 'username', 'password')) {
  print '<p>Successfully connected to MySQL!</p>';
  mysql_close($dbc);
} else {
  print '<p style="color: red;">Could not connect to MySQL.</p>';
}
```

通过将尝试连接的代码放到if-else语句的条件中，可以报告连接是否能够工作。

本章将继续使用username和password作为用户名和密码。对于所编写的脚本，需要使用Web主机提供的值来替换它们，或使用附录A中列出的步骤添加一个用户，并将用户名和密码设置到这里。

如果连接成功建立，则会打印一个正确消息并关闭连接。否则，会打印一个错误消息，而无需关闭数据库连接（因为根本就没打开）。

(4) 完成PHP代码和HTML页面：

```
?>
</body>
</html>
```

(5) 将文件保存为mysql_connect.php，放置在启用了PHP的计算机中适当的目录中，然后在Web浏览器中测试它（参见图12-4）。

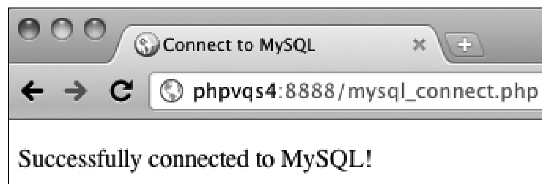


图12-4　如果PHP支持MySQL，并且使用的用户名/密码/主机这个组合是正确的，就会看到这个简单的消息

12

如果看到了类似图12-5中的结果，请再次检查用户名和密码值。这些值必须和所用的Web主机提供的值或在创建用户时使用的值一致。也可以在MySQL客户端（参见附录A）中测试连接用

的用户名和密码。



图12-5　如果PHP无法连接到MySQL，可能会看到类似这样的消息。错误消息可能出现也可能不出现，它取决于当前的错误管理设置

如果看到call to undefined function mysql_connect…这样的消息，可能是由于所用的PHP版本不支持MySQL（参见框注“PHP中对MySQL的支持”）。

PHP中对MySQL的支持

为了使用PHP中MySQL相关的函数，PHP必须内置对MySQL数据库服务器的支持。对于大多数PHP安装来说，这是默认支持的。你可以调用`phpinfo()`函数确认PHP是否支持MySQL，这个函数会详细显示PHP的安装信息。

在本章的学习过程中，如果看到了类似…undefined function mysql_…这样的错误消息，就表示使用的PHP版本不支持MySQL。（或者，也可能是拼写错了函数名，这也需要检查一下。）

启用MySQL支持只需要很少的操作，但却需要具备服务器的管理员级别权限。更多信息，请参见PHP手册。

✓提示

- ❑ 很多`mysql_`*`something`*`()`函数中的数据库连接参数（本例中是`$dbc`）都是可选的。但本书中的所有示例都没有省略它，就像它在`mysqli_`*`something`*`()`函数中不是可选参数一样。接下来你也会用到`mysqli_`*`something`*`()`函数，到时需要提供数据库连接（参见框注“MySQL扩展”）。

MySQL扩展

PHP可以通过两个不同的扩展与MySQL通信。首先是本章使用的标准MySQL扩展。该扩展已经使用了很多年，并且能够用于所有版本的PHP和MySQL。所有的标准MySQL扩展函数都以`mysql_`开头。

第二个扩展称作MySQLi（改进的MySQL扩展）。该扩展在PHP 5中添加，并能够用于

MySQL 4.1或更高版本。这些函数都以`mysqli_`开头，并且利用了MySQL中的一些新增功能。如果可能的话，最好使用MySQLi函数，但老版本的扩展应用得更为普遍，本书就只使用老版本。关于MySQLi扩展的详细信息，请参见PHP手册或我写的《PHP 6与MySQL 5基础教程》一书。

- 只有当PHP脚本和MySQL数据库位于同一台计算机上时，才能使用localhost作为主机名。通过修改PHP脚本中的主机名，并在MySQL中建立适当的权限，可以使用PHP连接到运行于远程服务器上的MySQL数据库。
- PHP对很多数据库（包括dBase、FilePro、mSQL、MySQL、Oracle、PostgreSQL和Sybase）都有内置的支持。如果使用的数据库类型不受直接支持（例如，Access或SQL Server），那么可以使用PHP的ODBC（开放数据库连接）函数以及数据库的ODBC驱动来与数据库交互。
- 现在组合使用PHP和MySQL非常常见，所使用的服务器可能同时配置了PHP和MySQL：LAMP、MAMP和WAMP。它们分别对应着不同的操作系统（Linux、Mac OS X和Windows）上的Apache Web服务器、MySQL DBMS以及PHP。
- 本章使用的是MySQL，所以用到的所有函数都是MySQL相关的。例如，要连接到MySQL中的数据库，可能使用`mysql_connect()`函数，但如果使用的是PostgreSQL，则需要使用`pg_connect()`来代替它。如果没有使用MySQL DBMS，请使用PHP手册（可以在www.PHP.net找到）查找对应的函数名。

12.3 MySQL 错误处理

本章在深入介绍如何使用MySQL之前，最好提前讨论一些用于处理错误的技术。会遇到的常见错误有：

- 连接MySQL失败；
- 选择数据库失败；
- 无法运行查询；
- 查询没有返回结果；
- 数据没有插入到表中。

根据经验，可以知道为什么通常会发生这些错误，但立即了解一下在运行脚本时发生了什么错误，便可以节省不少调试时间。要想通过脚本显示关于所发生错误的详实报告，请使用`mysql_error()`函数。该函数可以返回MySQL服务器返回关于错误的文本信息。

有了该函数，可能还需要使用一些PHP工具来处理错误。尤其是错误控制运算符（`@`），当在一个函数名之前使用该运算符时，可以阻止函数调用可能产生的任何错误消息或警告：

```
@function_name();
```

注意，该运算符并不能阻止错误的发生，它只是防止立即显示出错误消息。只有在发生错误

12

时希望自己处理时，才使用该运算符。

⇒ 使用错误处理

(1) 在文本编辑器或IDE中打开mysql_connect.php（参看脚本12-1）。

(2) 像下面这样（参看脚本12-2）修改if条件，阻止由mysql_connect()函数产生的任何PHP错误：

```
if ($dbc = @mysql_connect('localhost','username','password')) {
```

脚本12-2　通过向脚本中添加错误控制（@符号和mysql_error()函数），可以更有目的地处理发生的错误

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <title>Connect to MySQL</title>
7   </head>
8   <body>
9   <?php // 脚本12-2 - mysql_connect.php #2
10  /* 脚本连接到MySQL服务器。*/
11
12  // 尝试连接到MySQL并打印消息:
13  if ($dbc = @mysql_connect('localhost', 'username', 'password')) {
14
15    print '<p>Successfully connected to MySQL!</p>';
16
17    mysql_close($dbc); // 关闭连接。
18
19  } else {
20
21    print '<p style="color: red;">Could not connect to MySQL:<br/>'.mysql_error().'</p>';
22
23  }
24
25  ?>
26  </body>
27  </html>
```

这里使用@符号阻止了错误消息，当mysql_connect()函数发生意外时（参见图12-5），PHP不会打印出错误消息。错误仍然会发生，下一步的更改会处理它们。

(3) 在else部分中的print语句中添加mysql_error()函数：

```
print '<p style="color: red;">Could not connect to MySQL:<br/>' . mysql_error() . '.</p>';
```

该脚本不是打印错误消息或依赖PHP处理错误（参见图12-5），而是在当前上下文中打印MySQL错误。这是通过将一些HTML和mysql_error()函数连接在一起做到的。

注意mysql_error()函数，在这个例子中，没有提供数据库接连参数$dbc，因为没有创建数据库连接。

(4) 保存文件并在Web浏览器中再次测试（参见图12-6）。

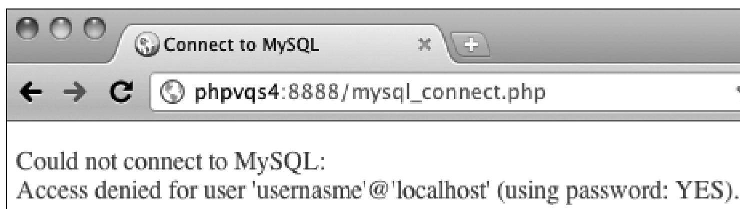


图12-6　使用PHP的错误控制函数，可以调整对错误进行处理的方式

如果发生了错误，其结果现在看起来要比图12-5好一些。如果脚本连接正常，结果看上去会是如图12-4所示的那样，因为此时并没有调用错误管理工具。

✓提示

- 在本章中，显示错误消息只是为了协助调试过程。真正的Web站点不应该将错误消息明确地显示给用户。
- 可以使用@符号阻止来自任何函数的错误、通知或警告，而不只是MySQL相关的函数。例如：

```
@include('./filename.php');
```

- 你可以看到，当发生连接错误时，还可以调用`die()`，这是`exit()`的同义词。其背后隐含的思想是，如果数据库连接无法建立，则程序不应继续运行。本章中省略了这些内容，因为这既笨重又容易使用不当。

12.4　创建和选择数据库

在PHP脚本能够与一个数据库进行交互之前，必须首先选中所需的数据库。当然，为了选中一个数据库，该数据库必须存在。可以使用PHP、MySQL客户端、phpMyAdmin或其他很多工具来创建数据库，只要使用的MySQL主机名/用户名/密码的组合具备这样做的权限。

数据库权限比文件权限要稍微复杂一些，但需要理解这样一个事实——不同类型的用户可以被赋予不同的数据库权限。例如，一个DBMS用户也许可以创建新数据库或删除现有数据库（所用的DBMS中可能有大量的数据库），但一个低级用户也许只能在一个单独的数据库中创建或修改表，而更多的基本用户也许只能从表中读取数据，但不能修改表。

如果正在托管站点中使用PHP和MySQL，则主机托管公司通常会给予你第二种类型的访问——控制一个单独的数据库而不是DBMS本身——并建立好初始数据库。如果正在使用自己的服务器，或者具备管理员权限，则可以创建新用户和数据库。

要使用PHP创建数据库，可以使用`mysql_query()`函数结合`CREATE DATABASE` *`databasename`* SQL命令：

```
mysql_query('CREATE DATABASE somedb', $dbc);
```

当创建完数据库后，可以使用mysql_select_db()函数来选中该数据库：

```
mysql_select_db('somedb', $dbc);
```

注意，任何时候一个数据库只需要创建一次，但每次在其上运行查询之前都需要选中它。换句话说，一些开发人员需要执行前述的第一步（指创建数据库），而每个人在每个PHP脚本中都需要执行前述第二步（指选中数据库）。

在这个例子中，将创建一个新的数据库并选中它。再次重申，创建数据库的查询需要具备管理员访问权限。如果Web主机对访问进行阻止，则它应该为后续请求创建初始数据库，这时只需编写该脚本的第二部分——选择数据库。

⇛ 创建并选中数据库

(1) 在文本编辑器或IDE中打开mysql_connect.php（参看脚本12-2）。

(2) 如果需要的话，在第一个print语句后创建新的数据库（参看脚本12-3）：

```
if (@mysql_query('CREATE DATABASE myblog', $dbc)) {
  print '<p>The database has been created!</p>';
} else {
  print '<p style="color: red;">Could not create the database because:<br/>' .
  →mysql_error($dbc) . '.</p>');
}
```

脚本12-3　通过3步创建了一个新的数据库：连接到数据库，使用mysql_query()函数运行CREATE DATABASE查询，然后关闭连接

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Create the Database</title>
7   </head>
8   <body>
9   <?php // 脚本12-3 - create_db.php
10  /* 脚本连接到MySQL服务器。创建并选中数据库。*/
11
12  // 尝试连接到MySQL并打印消息：
13  if ($dbc = @mysql_connect('localhost', 'username', 'password')) {
14
15     print '<p>Successfully connected to MySQL!</p>';
16
17     // 尝试创建数据库：
18     if (@mysql_query('CREATE DATABASE myblog', $dbc)) {
19        print '<p>The database has been created!</p>';
20     } else { // 无法创建数据库。
21        print '<p style="color: red;">Could not create the database because:<br/>' .
          mysql_error($dbc) . '.</p>';
22     }
23
24     // 尝试选中数据库：
```

```
25     if (@mysql_select_db('myblog', $dbc)) {
26        print '<p>The database has been selected!</p>';
27     } else {
28        print '<p style="color: red;">Could not select the database because:<br/>' .
          mysql_error($dbc) . '.</p>';
29     }
30
31     mysql_close($dbc); // 关闭连接。
32
33  } else {
34
35     print '<p style="color: red;">Could not connect to MySQL:<br/>' . mysql_error() . '.</p>';
36
37  }
38
39  ?>
40  </body>
41  </html>
```

如果需要创建数据库，使用该构造来清晰有效地完成这一任务。使用`mysql_query()`函数来运行`CREATE DATABASE myblog`这个查询。`@`符号用于阻止错误消息，转而在`else`子句中通过连接`print`和`mysql_error()`函数来处理该错误。

注意，这个`mysql_error()`调用可以提供特定的数据库连接参数：`$dbc`。

如果已经创建了该数据库，请跳过这一步。

(3) 尝试选中该数据库：

```
if (@mysql_select_db('myblog', $dbc)) {
  print '<p>The database has been selected!</p>';
} else {
  print '<p style="color: red;"> Could not select the database because:<br/>' .
  →mysql_error($dbc) . '.</p>';
}
```

这个条件语句和第2步中的结构一样。如果PHP能够选中数据库，则会打印一条消息。如果不能选中该数据库，则会转而显示特定的MySQL错误。

PHP在数据库上运行查询的任何脚本都必须在连接到MySQL并选中数据库后才能工作。

(4) 如果需要的话，可以修改页面的标题来反映脚本的新目的：

```
<title>Create the Database</title>
```

(5) 将脚本保存为`create_db.php`，放置在启用了PHP的服务器中适当的目录中，然后在Web浏览器中测试（参见图12-7和图12-8）。

Successfully connected to MySQL!

The database has been created!

The database has been selected!

图12-7 如果数据库可以创建并选中，将会在浏览器中看到这样的结果

```
Successfully connected to MySQL!

Could not create the database because:
Access denied for user 'username'@'localhost' to database 'myblog'.

Could not select the database because:
Unknown database 'myblog'.
```

图12-8　如果用户没有授权创建数据库，将会看到这样的消息。如果没有权限选中数据库，也会出现类似的结果

✓提示

- 可能不需要频繁地创建数据库，而且通常也不会使用PHP脚本来创建数据库。不过，该实例同时演示了如何使用PHP执行简单的查询和创建数据库所需的SQL命令。
- 通常，在设置数据库信息（主机名、用户名、密码和数据库名称）时，最好使用变量或常量，但这些例子里并没有这么做。使用变量或常量，可以将这些代码插入到适当的函数中。这样做还可以将数据库特定的代码同脚本的功能性代码分离开，易于将这些代码移植到其他应用程序中。

12.5　创建表

在创建并选中了初始数据库之后，就可以开始在其中创建单独的表了。一个数据库可以由多个表构成，但在这个实例中只创建一个用于存放数据的表。

要在数据库中创建表，需要使用SQL这种数据库能够理解的语言。因为SQL非常像英语口语，所以创建新表的查询如下所示：

```
CREATE TABLE tablename (column1 definition, column2 definition, etc.)
```

每个列之间用逗号分隔，首先指定列的名字，然后是列类型。常见的类型有`TEXT`、`VARCHAR`（变长字符串）、`DATETIME`和`INT`（整数）。

强烈建议将第一个列创建为**主键**（primary key）（用于引用整个行的列），因此一个简单的`CREATE`语句是：

```
CREATE TABLE my_table (
id INT PRIMARY KEY,
information TEXT
)
```

表的主键是一个特殊列，由用于引用表中整个行的唯一值构成。数据库会为该列创建一个索引，以便更快地在表中导航。一个表只能有一个主键，通常将其设置为自动增长的整数列。其第一行的键值是`1`，第二行的键值是`2`，依此类推。引用键值永远都可以检索到对应行的数据。

可以访问MySQL网站查看关于SQL的更多信息。然而，按照这一节的介绍，将能够完成基本的数据库任务。表12-2描述了在这个示例中将要创建的表。

表12-2 **Entries**表

列 名 称	列 类 型
entry_id	自动增长的正整数，非空
title	最长为100字符的文本
entry	任意长度的文本
date_entered	包含了该行添加日期和时间的时间戳

在这个例子中，将要创建一个表，用于存放通过HTML表单提交的信息。在12.6节中，还将编写脚本，将提交的数据插入到这里创建的表中。

⇛ 创建一个新表

(1) 在文本编辑器或IDE中打开新的PHP文档，命名为create_table.php（参看脚本12-4）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Create a Table</title>
</head>
<body>
```

脚本12-4 要创建数据库表，首先定义适当的SQL语句，然后调用mysql_query()函数

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Create a Table</title>
7   </head>
8   <body>
9   <?php // 脚本12-4 - create_table.php
10  /* 脚本连接到MySQL服务器。选中数据库并创建数据库表。*/
11
12  // 连接服务器并选中数据库：
13  if ($dbc = @mysql_connect('localhost', 'username', 'password')) {
14
15     // 如果数据库无法选中，打印错误信息：
16     if (!@mysql_select_db('myblog', $dbc)) {
17        print '<p style="color: red;">Could not select the database because:<br/>' .
          mysql_error($dbc) . '.</p>';
18        mysql_close($dbc);
19        $dbc = FALSE;
20     }
21
22  } else { // 连接失败。
23     print '<p style="color: red;">Could not connect to MySQL:<br/>' . mysql_error() . '.</p>';
24  }
25
26  if ($dbc) {
27
28     // 定义查询：
```

```
29    $query = 'CREATE TABLE entries (
30  entry_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
31  title VARCHAR(100) NOT NULL,
32  entry TEXT NOT NULL,
33  date_entered DATETIME NOT NULL
34  )';
35
36    // 执行查询:
37    if (@mysql_query($query, $dbc)) {
38       print '<p>The table has been created!</p>';
39    } else {
40       print '<p style="color: red;"> Could not create the table because:<br/>' .
         mysql_error($dbc) . '.</p> <p>The query being run was: ' . $query . '</p>';
41    }
42
43    mysql_close($dbc); // 关闭连接。
44
45  }
46  ?>
47  </body>
48  </html>
```

(2) 打开一个PHP代码片段：

```
<?php // 脚本12-4 - create_table.php
```

(3) 连接到MySQL服务器并选中数据库：

```
if ($dbc = @mysql_connect ('localhost', 'username', 'password')) {
  if (!@mysql_select_db('myblog', $dbc)) {
    print '<p style="color: red;">Could not select the database because:<br/>' .
    →mysql_error($dbc) . '.</p>';
    mysql_close($dbc);
    $dbc = FALSE;
  }
} else {
  print '<p style="color: red;">Could not connect to MySQL:<br/>' . mysql_error()
  →.'.</p>';
}
```

这是前面脚本中使用的代码的另一个版本。主要的区别在于如果每一步都成功了，则不会打印任何消息（这里我们假设所有代码都能正常工作）。

如果出于某些原因，无法连接或选中数据库，则会打印错误信息（参见图12-9）。这样做是有意义的，因为如果数据库无法选中，就不能尝试在其中创建表。表示连接的`$dbc`变量会被设置为FALSE，指示不要执行赋予其的`CREATE`查询（参见第(4)步）。

(4) 创建查询来建立表：

```
if ($dbc) {
  $query = 'CREATE TABLE entries (
entry_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
title VARCHAR(100) NOT NULL,
entry TEXT NOT NULL,
date_entered DATETIME NOT NULL
)';
```



图12-9 根据MySQL错误消息和打印出来的正在执行的查询语句，
应该能够找出导致脚本无法正确执行的错误

首先，如果$dbc具有值，则可以创建表，如果没有值，则意味着链接无法建立或数据库无法选中，这样之后的代码都不应执行。对于查询本身，我们可以将其分割成易于识别的几个部分。首先，要创建新表，应写下CREATE TABLE *tablename*（其中*tablename*应该用实际所需的表名称替换掉）。然后，在括号中列出所需的每一列，列之间用逗号分开。表和列的名字应该是字母或数字，不能有空格。

该表的第一列称作entry_id，这是一个无符号整数（INT UNSIGNED，表示它只能存放正整数）。通过添加单词NOT NULL，可以指定该列对于每一行都必须具有一个值。其值在每次插入新行时自动增加（AUTO INCREMENT），并用作主键。

接下来的两列由文本构成。其中一个名为title，被限制为100字符。第二个名为entry，可以拥有几乎无限的大小。这两个字段也都被标记为NOT NULL，表示它们是必需的字段。

最后，date_entered列是一个时间戳，标志着每一条记录是何时被添加到表中的。

(5) 执行查询：

```
if (@mysql_query($query, $dbc)) {
  print '<p>The table has been created.</p>';
} else {
  print '<p style="color: red;">Could not create the table because:<br/>' .
  →mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
}
```

创建表，使用$query变量作为参数来调用mysql_query()函数。如果发生错误，会打印MySQL错误以及$query变量的值。最后这一步（打印正在执行的查询语句）是非常有用的调试技术（参见图12-10）。

Could not create the table because:
CREATE command denied to user 'username'@'localhost' for table 'entries'.

The query being run was: CREATE TABLE entries (entry_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY, title VARCHAR(100) NOT NULL, entry TEXT NOT NULL, date_entered DATETIME NOT NULL)

图12-10 如果查询发生错误，会报告MySQL错误，并显示出错的查询语句（出于调试目的）

(6) 关闭数据库连接并完成$dbc条件语句：

```
  mysql_close($dbc);
}
```

(7) 完成PHP代码和HTML页面：

```
?>
</body>
</html>
```

(8) 将脚本保存为create_table.php，放置在启用了PHP的服务器上的适当目录中，并在Web浏览器中测试（参见图12-11）。



图12-11　如果一切正常，则会看到该信息

✓提示

- 编写SQL查询时并不一定要像这里这样全部使用全大写字母，但这样做有助于区分SQL关键字与表和列的名字。
- 在大型Web应用程序中，强烈建议将数据库连接和选中代码（这里的第13～24行）放置在单独的文件中，并放置在Web目录之外。然后，每个需要查询数据库的页面都来包含这个外部文件。
- 只要查询在数据库上运行成功了，mysql_query()函数就会返回TRUE。但这并不意味着期望的结果就必定发生了。
- 本章只讲述了MySQL和SQL相关的基本知识（包括列类型）。当对这些基础知识掌握之后，可能希望查看其他资源，这些在附录B中都列出了。
- 一般不需要使用PHP脚本创建表，就像一般不需要使用PHP脚本创建数据库一样。但是当刚开始使用MySQL时，这是完成所需工作的简单方式

12.6　向数据库插入数据

前面已经提到过，该数据库将用于一个博客，这是一种在线的日记。博客条目（由标题和文本构成）将通过一个页面添加到数据库中，然后显示在另一个页面中。这是一个简单的示例，但与数据库的使用关系紧密。

在12.5节中，我们创建了一个表，它由4列构成：entry_id、title、entry和date_entered。从所使用的函数来看，向表中插入信息的过程和创建表的过程是类似的，但是用的SQL查询是不

同的。要插入记录，可以通过下面这两种模式来使用INSERT SQL语法：

```
INSERT INTO tablename VALUES (value1, value2, value3, etc.)
INSERT INTO tablename (column1_name, column2_name) VALUES (value1, value2)
```

该查询以INSERT INTO *tablename*开始。然后可以指定要向哪些列插入值（也可以不指定）。后面一种形式更为明确，因此我也更加推荐这种形式，但当插入大量列时，这种形式更加冗长。不管采用哪种形式，必须为表中的每一行列出正确数量的列和值的类型。

值放在括号中，每个值之间用逗号分隔。字符串和日期这样的非数字值需要使用引号，数字值则不用：

```
INSERT INTO example (name, age) VALUES ('Jonah', 1)
```

使用mysql_query()函数可以在数据库上运行该查询。由于INSERT查询通常都很复杂，因此有必要给每个查询都赋予一个变量，然后将该变量传递给mysql_query()函数（和前面的示例一样）。

作为演示，我们来创建一个页面将blog条目添加到数据库中。和之前章节中的例子一样，该页面同时显示和处理HTML表单。然而，在进入这个示例之前，要注意该脚本包含安全漏洞，我们将在本章的12.7节中解释并修复该漏洞。

⇛ 通过HTML表单将数据插入到数据库中

(1) 在文本编辑器或IDE中打开一个新的PHP文档，命名为add_entry.php（参看脚本12-5）。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Add a Blog Entry</title>
</head>
<body>
<h1>Add a Blog Entry</h1>
```

脚本12-5 向数据库中添加信息的查询语句非常简单，但要确保括号中值的数量和数据库表中列的数量是匹配的

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Add a Blog Entry</title>
7   </head>
8   <body>
9   <h1>Add a Blog Entry</h1>
10  <?php // 脚本12-5 - add_entry.php
11  /* This script adds a blog entry to the database. */
12
13  if ($_SERVER['REQUEST_METHOD'] == 'POST') { // 处理表单。
```

```
14
15      // 连接服务器并选中数据库：
16      $dbc = mysql_connect('localhost', 'username', 'password');
17      mysql_select_db('myblog', $dbc);
18
19      // 验证表单数据：
20      $problem = FALSE;
21      if (!empty($_POST['title']) && !empty($_POST['entry'])) {
22         $title = trim(strip_tags($_POST['title']));
23         $entry = trim(strip_tags($_POST['entry']));
24      } else {
25         print '<p style="color: red;">Please submit both a title and an entry.</p>';
26         $problem = TRUE;
27      }
28
29      if (!$problem) {
30
31         // 定义查询：
32         $query = "INSERT INTO entries (entry_id, title, entry, date_entered) VALUES
           (0, '$title', '$entry', NOW())";
33
34         // 执行查询：
35         if (@mysql_query($query, $dbc)) {
36            print '<p>The blog entry has been added!</p>';
37         } else {
38            print '<p style="color: red;">Could not add the entry because:<br/>' .
              mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
39         }
40
41     } // 一切正常！
42
43     mysql_close($dbc); // 关闭连接。
44
45   } // 结束提交条件语句。
46
47   // 显示表单：
48   ?>
49   <form action="add_entry.php" method="post">
50      <p>Entry Title: <input type="text" name="title" size="40" maxsize="100" /></p>
51      <p>Entry Text: <textarea name="entry" cols="40" rows="5"></textarea></p>
52      <input type="submit" name="submit" value="Post This Entry!" />
53   </form>
54   </body>
55   </html>
```

(2) 创建初始PHP段，并检查是否正在提交表单：

```
<?php // 脚本12-5 - add_entry.php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

(3) 连接并选中数据库：

```
$dbc = mysql_connect('localhost', 'username', 'password');
mysql_select_db('myblog', $dbc);
```

此时，如果正在按顺序运行这些示例，本书将假设已经有了能够工作的连接和选中过程，因此省略了所有的条件语句和错误报告（极大地缩短了脚本）。如果在连接和选中数据库时遇到了问题，请使用本章之前列出的代码。

(4) 验证表单数据：

```
$problem = FALSE;
if (!empty($_POST['title']) && !empty($_POST['entry'])) {
  $title = trim(strip_tags ($_POST['title']));
  $entry = trim(strip_tags ($_POST['entry']));
} else {
  print '<p style="color: red;">Please submit both a title and an entry.</p>';
  $problem = TRUE;
}
```

在INSERT查询中使用表单数据之前，应该对其进行验证。这里只使用了最少的验证，确保确实提供了某些值。如果验证成功，这些值会在截去两端的空格后，赋给新的变量。如果没有通过验证，会打印错误消息（参见图12-12）并将$problem标志变量设置为TRUE（因为发生了错误）。

图12-12 PHP还是执行了一些基本的表单验证，因此空的记录是不会插入到数据库中的

(5) 定义INSERT查询：

```
if (!$problem) {
  $query = "INSERT INTO entries (entry_id, title, entry, date_entered) VALUES
  →(0, '$title', '$entry', NOW())";
```

该查询由必需的INSERT INTO *tablename*代码开始。然后列出了与提交的值对应的列。之后是单词VALUES，后跟4个值（每个列一个，按顺序放置）放置在单引号中并用逗号分开。当把该查询赋值给$query变量时，要使用双引号这样变量的值就会被PHP自动插入。$title和$entry变量是字符串，因此在查询中要用引号括起来。

由于entry_id列被设置为AUTO_INCREMENT，因此可以使用0作为其值，MySQL会自动为该列使用逻辑上的下一个值。要设置date_entered列的值，可以使用MySQL函数NOW()。它会将当前时间作为值插入。

(6) 在数据库上运行查询：

```
if (@mysql_query($query, $dbc)) {
  print '<p>The blog entry has been added!</p>';
} else {
  print '<p style="color: red;"> Could not add the entry because:<br/>' . mysql_
  →error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
}
```

一旦定义好查询，就可以使用mysql_query()函数运行它。将对该函数的调用作为if-else语句的条件，通过这种方式可以基于查询执行的结果打印简单的消息。

由于这本身就是一个调试工具，所以当查询没有正确运行时，会在Web浏览器中打印出MySQL错误和正在运行的查询（参见图12-13）。

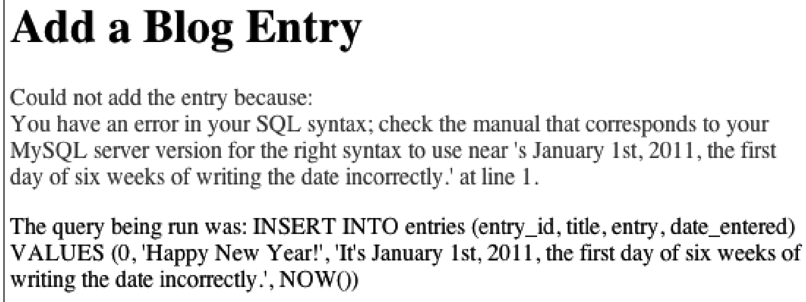


图12-13　如果INSERT查询不能工作，会打印出MySQL错误和正在运行的查询

(7) 关闭$problem条件语句、数据库连接，完成主条件语句和PHP片段：

```
  } // 一切正常！
  mysql_close($dbc);
} // 结束提交条件语句。
?>
```

从这里开始，表单将会被显示出来。

(8) 创建表单：

```
<form action="add_entry.php" method="post">
  <p>Entry Title: <input type="text" name="title" size="40" maxsize="100" /></p>
  <p>Entry Text: <textarea name="entry" cols="40" rows="5"></textarea></p>
  <input type="submit" name="submit" value="Post This Entry!" />
</form>
```

这个HTML表单非常简单，用于请求博客的标题和内容。一个很好的经验规则是，为表单的输入框使用的名字应该和数据库中的列名字对应起来。这样做可以降低错误发生的几率。

(9) 完成HTML页面：

```
</body>
</html>
```

(10) 将脚本保存为add_entry.php，放置在启用了PHP的服务器上适当的目录中，然后在

Web浏览器中测试（参见图12-14和图12-15）。

图12-14　用于向数据库中添加条目的表单

图12-15　如果INSERT查询正确地运行了，会打印一条消息并再次显示表单

在表单的值中应该避免使用撇号，否则将会看到如图12-13所示的结果。12.7节会详细介绍此问题并给出解决方法。

✓提示

- MySQL支持使用下面的格式一次插入多条记录：

```
INSERT INTO tablename (column1_name, column2_name) VALUES (value1, value2),
→(value3, value4);
```

  然而，大多数其他数据库应用程序不支持这种结构。

- 要检索为AUTO_INCREMENT列自动产生的数值，可以使用mysql_insert_id()函数。
- 由于主键自动递增，查询也可以写成：

```
INSERT INTO entries (title, entry, date_entered) VALUES ('$title', '$entry', NOW())";
```

创建示例

本章的重点在于诠释PHP与MySQL的基础应用，同时也会介绍SQL的核心组件。但是，示例中演示的内容一般不会用在实际的网站中，比如允许任何人插入、编辑、删除数据库记录。

在第13章，我们会开发一个完全不同的示例，该示例也是由数据库驱动的。示例会使用cookie并限制用户在网站上的行为。

12.7　安全查询数据

在介绍之前的步骤时，本书已经提到过，这里编写的代码中存在着很恶劣的安全漏洞。对于这样的代码，如果某些用户提交的文本中包含一个撇号，这样的数据就会破坏SQL查询（参见图

12-16)。这个结果很明显不是我们想要的，但是为什么这是不安全的呢?

如果一个恶意用户知道他可以通过输入一个撇号来破坏查询，那么他就可以试图利用该漏洞运行他自己的查询。如果有的用户提交';DROP TABLE entries;作为blog条目的标题，则产生的查询将是：

```
INSERT INTO entries (entry_id, title, entry, date_entered) VALUES
→(0, '';DROP TABLE entries;', '<entry text>', NOW())
```

Add a Blog Entry

Could not add the entry because:
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ';DROP TABLE entries;', '', NOW())' at line 1.

The query being run was: INSERT INTO entries (entry_id, title, entry, date_entered) VALUES (0, '';DROP TABLE entries;', '', NOW())

图12-16　连接时的撇号会破坏查询，因为撇号（或单引号）用于分隔查询中使用的字符串

第一个撇号完成了查询中博客标题的值。接下来，分号结束了INSERT查询语句。这导致原本的查询语句发生语法错误。接下来，向数据库提交了第二个查询语句——DROP TABLE entries，这个语句顺利执行，而原来的INSERT查询会失败。这被称作SQL*注入攻击*（SQL injection attack），但幸运的是这很容易避免。

要避免这样的问题，在向查询传递可能不安全的数据时，请使用mysql_real_escape_string()函数。该函数会对任何可能危险的字符进行转义，即在其前面添加一个反斜线，确保查询中使用的数据是安全的：

```
$var = mysql_real_escape_string($var, $dbc);
```

下面我们对前面的脚本应用该函数。

⇛ 使查询数据更安全

(1) 如果还没打开add_entry.php脚本（参看脚本12-5），在文本编辑器或IDE中打开它。

(2) 更新所指派的$title和$entry变量以便读取（参看脚本12-6）：

```
$title = mysql_real_escape_string(trim(strip_tags($_POST['title'])), $dbc);
$entry = mysql_real_escape_string(trim(strip_tags($_POST['entry'])), $dbc);
```

脚本12-6　要使Web应用程序和数据库更为安全，需要针对在查询中使用的表单数据应用mysql_real_escape_string()函数

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Add a Blog Entry</title>
```

```
7   </head>
8   <body>
9   <h1>Add a Blog Entry</h1>
10  <?php // 脚本12-6 - add_entry.php #2
11  /* 脚本向数据库中添加一篇博客，并使查询更加安全。*/
12
13  if (isset($_POST['submitted'])) { // 处理表单。
14
15    // 连接服务器并选中数据库:
16    $dbc = mysql_connect('localhost', 'username', 'password');
17    mysql_select_db('myblog', $dbc);
18
19    // 验证表单数据并确保安全:
20    $problem = FALSE;
21    if (!empty($_POST['title']) && !empty($_POST['entry'])) {
22       $title = mysql_real_escape_string(trim(strip_tags($_POST['title'])), $dbc);
23       $entry = mysql_real_escape_string(trim(strip_tags($_POST['entry'])), $dbc);
24    } else {
25       print '<p style="color: red;">Please submit both a title and an entry.</p>';
26       $problem = TRUE;
27    }
28
29    if (!$problem) {
30
31       // 定义查询:
32       $query = "INSERT INTO entries (entry_id, title, entry, date_entered) VALUES
         (0, '$title', '$entry', NOW())";
33
34       // 执行查询:
35       if (@mysql_query($query, $dbc)) {
36          print '<p>The blog entry has been added!</p>';
37       } else {
38          print '<p style="color: red;">Could not add the entry because:<br/>' .
            mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
39       }
40
41    } // 一切正常!
42
43     mysql_close($dbc); // 关闭连接。
44
45  } // 结束提交条件语句。
46
47  // 显示表单:
48  ?>
49  <form action="add_entry.php" method="post">
50     <p>Entry Title: <input type="text" name="title" size="40" maxsize="100" /></p>
51     <p>Entry Text: <textarea name="entry" cols="40" rows="5"></textarea></p>
52     <input type="submit" name="submit" value="Post This Entry!" />
53     <input type="hidden" name="submitted" value="true" />
54  </form>
55  </body>
56  </html>
```

这两行会大大改善脚本的安全性和功能。提交的两个变量，首先截去了两端的空格，然后被

传递给了mysql_real_escape_string()。其结果可以安全地用于查询。

如果你觉得上面的代码（将3个函数的返回值赋给同一个变量）不太容易理解，可以将代码分开写成3条语句：

```
$title = $_POST['title'];
$title = trim(strip_tags($title));
$title = mysql_real_escape_string($title, $dbc);
```

(3) 保存脚本，将其放置在启用了PHP的服务器上的适当目录中，然后在Web浏览器中测试它（参见图12-17和图12-18）。

Add a Blog Entry

Entry Title: It's Another Test!

Entry Text: "Will these quotes and apostrophes cause problems?", you ask. I don't think so!

Post This Entry!

图12-17　现在再在表单中输入撇号

Add a Blog Entry

The blog entry has been added!

图12-18　已经不会产生问题了

✓提示

❑（在本章稍后的部分）如果看到显示出来的博客条目在撇号之前有额外的反斜线，则很可能是使用了PHP版本6之前的版本，并启用了Magic Quotes。（即便没有使用mysql_real_escape_string()，Magic Quotes也会自动转义表单数据中有问题的字符。）如果是这种情况，需要使用stripslashes()函数从提交的值中移除额外的反斜线：

```
$title = mysql_real_escape_string(stripslashes(trim(strip_tags($_POST['title']))), $dbc);
```

显示MySQL错误

即使MySQL没有执行注入的SQL命令（通常MySQL只会运行mysql_query()函数发送的一条SQL查询语句），但是黑客会在表单数据中提交一些恶意的字符，破坏查询语句的语法，从而产生数据库错误。通过查看数据库错误，黑客就会获得相关数据库的信息，通过这些信息他们就可以达到各种各样的恶意目的。出于以上原因，一定不要在真实的网站上显示MySQL错误或正在执行的查询语句。本章脚本中使用这些语句是出于调试代码的目的。

12.8　从数据库中检索数据

本章要演示的操作数据库的下一个步骤是从已有的表中检索数据。这里依然使用mysql_query()函数，但检索数据和插入数据有些不同：必须将检索到的信息赋给一个变量，然后使用另一个函数获取数据。

检索数据的基本语法是SELECT查询：

```
SELECT what columns FROM what table
```

从一个表中读取数据，最简单的查询是：

```
SELECT * FROM tablename
```

其中的星号表示查询所有列。如果只需要返回某几列，可以对查询作出限定，如：

```
SELECT name, email FROM users
```

这个查询要求只生成两列（name和email）信息。要记住，这种结构不会限制返回哪些行（或记录），只是返回所有行的某些列。

该查询的另一种修改方式是添加一个条件，限制返回哪些行，这通过WHERE子句来完成：

```
SELECT * FROM users WHERE name='Larry
```

这里需要的是表中所有列的信息，但只返回那些name列等于Larry的行。

这是一个很好的示例，当SQL只需要少量条目时，这样做很有效而且很灵活。

从数据库中检索数据和向数据库中插入数据的主要区别在于，需要对查询进行不同的处理。需要将查询的结果赋给一个变量：

```
$result = mysql_query($query, $dbc);
```

$dbc是一个打开的数据库连接的引用，而$result是一个查询结果集的引用。$result变量接下来会提交给mysql_fetch_array()函数，以获取查询结果：

```
$row = mysql_fetch_array($result);
```

函数从结果集中每次获取一行数据，并在这一过程中创建一个数组。这个数组会使用已选择的列名作为索引：$row['name']、$row['email']等。和使用其他数组一样，必须通过数据库中定义的确切列名来引用其中的列（列名是区分大小写的）。因此，在这个例子中，必须使用$row['email']，而不能使用$row['Email']。

如果查询会返回多行数据，在循环中执行mysql_fetch_array()函数，以获取所有的行：

```
while ($row = mysql_fetch_array($result)) {
        // Do something with $row.
}
```

在循环的每次迭代中，来自查询结果（通过$result引用）的下一行信息会被转移到一个称作$row的数组中。这一过程一直持续，直到从返回结果中再也找不到任何行信息为止。在循环内部，可以对$row执行任何操作。

了解这个系统的最佳方式就是去尝试它。接下来我们将编写一个脚本，检索存放在entries表中的条目并显示它们（参见图12-19）。之前应该已经创建了这个表，并且不止一次地运行了add_entry.php。

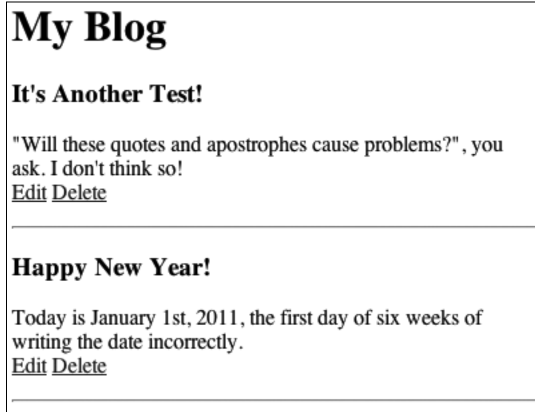


图12-19　这个动态Web页面使用PHP从数据库中提取数据

⇛ 从表中检索数据

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为view_entries.php（参看脚本12-7）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>View My Blog</title>
</head>
<body>
  <h1>My Blog</h1>
```

脚本12-7　用于从一个表中检索所有数据的SQL查询非常简单，但为了让PHP能够访问返回的每一行记录，需要在循环中每次访问结果中的一行

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>View My Blog</title>
7   </head>
8   <body>
9      <h1>My Blog</h1>
10  <?php // 脚本12-7 - view_entries.php
11  /* This script retrieves blog entries from the database. */
12
13  // 连接服务器并选中数据库:
14  $dbc = mysql_connect('localhost', 'username', 'password');
15  mysql_select_db('myblog', $dbc);
16
```

```
17  // 定义查询:
18  $query = 'SELECT * FROM entries ORDER BY date_entered DESC';
19
20  if ($r = mysql_query($query, $dbc)) { // 运行查询。
21
22     // 打印返回值:
23     while ($row = mysql_fetch_array($r)) {
24        print "<p><h3>{$row['title']}</h3>
25        {$row['entry']}<br/>
26        <a href=\"edit_entry.php?id={$row['entry_id']}\">Edit</a>
27        <a href=\"delete_entry.php?id={$row['entry_id']}\">Delete</a>
28        </p><hr />\n";
29     }
30
31  } else { // 没有运行查询。
32    print '<p style="color: red;">Could not retrieve the data because:<br/>' .
      mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
33  } // 结束查询条件语句。
34
35  mysql_close($dbc); // 关闭连接。
36
37  ?>
38  </body>
39  </html>
```

(2) 定义PHP片段并连接到数据库：

```
<?php // 脚本12-7 - view_entries.php
$dbc = mysql_connect('localhost', 'username', 'password');
mysql_select_db('myblog', $dbc);
```

(3) 定义SELECT查询：

```
$query = 'SELECT * FROM entries ORDER BY date_entered DESC';
```

该基本查询告诉数据库需要查看entries表中所有行所有列的数据。返回的结果应该进行排序，这通过ORDER BY子句指定了按照输入的时间（记录在date_entered列中）进行排序，最先显示最近插入的记录。语句最后的DESC选项是descending的简写。如果将查询改为ORDER BY date_entered ASC，则最新插入的记录将被排在最后。

(4) 运行查询：

```
if ($r = mysql_query($query, $dbc)) {
```

SELECT查询和其他查询的运行方式一样。然而，查询的结果需要赋给$result（或者更短些，$r）变量，以供之后引用。

(5) 打印返回的值：

```
while ($row = mysql_fetch_array($r)) {
  print "<p><h3>{$row['title']}</h3>
  {$row['entry']}<br/>
  <a href=\"edit_entry.php?id={$row['entry_id']}\">Edit</a>
  <a href=\"delete_entry.php? id={$row['entry_id']}\"> Delete</a>
  </p><hr />\n";
}
```

该循环将包含有$r中返回的第一条记录的数组赋给$row变量。然后循环会执行下面的命令（print语句）。每当循环回到起点，如果还有其他行的话，它会再次将下一行赋值给$row。如此继续下去，直到无法再获取更多行信息为止。

数组的键就是表中列的名字，也就是entry_id、title和entry（这里无需打印出date_entered）。

在每个条目的底部，还创建了两个链接：指向edit_entry.php和delete_entry.php。这些功能将在本章其他部分中编写。每个链接都将条目在数据库中的ID值传递到URL中。在另外两个页面对blog条目进行编辑和删除时，该信息是必不可少的。

(6) 处理查询无法执行的错误：

```
} else { // 没有运行查询。
  print '<p style="color: red;">Could not retrieve the data because:<br/>' .
  →mysql_error($dbc) . '.</p> <p>The query being run was: ' . $query . '</p>';
} // 结束查询条件语句。
```

如果无法在数据库上运行查询，则应该打印出该查询，以及MySQL错误（用于调试）。

(7) 关闭数据库连接：

```
mysql_close($dbc);
```

(8) 完成PHP片段和HTML页面：

```
?>
</body>
</html>
```

(9) 将脚本保存为view_blog.php，放置在启用了PHP的服务器上的适当目录中，然后在Web浏览器中进行测试（参见图12-19）。

(10) 如果需要的话，可以使用add_entry.php页（参看脚本12-6）添加另一条博客记录，并再次运行该页面（参见图12-20）。



图12-20　感谢SELECT查询，将返回的记录按照输入的结果进行了排序，使最新添加的记录总是列在最前面

(11) 如果希望的话，检查页面的源代码，查看动态生成的链接（参见图12-21）。

```
        <h1>My Blog</h1>
<p><h3>This is the newest post!</h3>
                This is so absolutely amazing that I'm dowr
                <a href="edit_entry.php?id=4">Edit</a>
                <a href="delete_entry.php?id=4">Delete</a>
                </p><hr />
<p><h3>It's Another Test!</h3>
                "Will these quotes and apostrophes cause pr
so!<br />
                <a href="edit_entry.php?id=3">Edit</a>
                <a href="delete_entry.php?id=3">Delete</a>
                </p><hr />
<p><h3>Happy New Year!</h3>
                Today is January 1st, 2011, the first day c
incorrectly.<br />
                <a href="edit_entry.php?id=2">Edit</a>
                <a href="delete_entry.php?id=2">Delete</a>
                </p><hr />
```

图12-21　页面HTML源代码的一部分。注意，两个链接都在URL后面追加了?id=x

✓提示

- mysql_fetch_array()函数还可以接受另一个参数，这是一个常量，它指定了需要返回哪种类型的数组。MYSQL_ASSOC返回关联数组，而MYSQL_NUM返回按数值索引的数组。
- mysql_num_rows()函数返回由SELECT查询返回的记录行数。
- 还可以分页返回数据，如每页返回10条或20条记录（就像Google那样）。但这样做需要编写更多更高级代码，这已经超出了本书的讨论范围。请参考我写的《PHP6与MySQL基础教程》，或在线查找一些代码示例和教程。

12.9　删除数据库中的数据

有的时候可能需要在数据库上运行DELETE查询。该查询用于从数据库中移除记录。DELETE查询的语法是：

```
DELETE FROM tablename WHERE column=value
```

其中的WHERE子句并不是必需的，但如果省略它，将会从表中移除所有记录。还要记得，一旦删除了一条记录，就无法恢复它（除非拥有数据库的备份）。

作为防范措施，如果你希望从表中只删除一条记录，就可以为查询指定LIMIT子句：

```
DELETE FROM tablename WHERE column=value LIMIT 1
```

该子句确保了最多只有一条记录会被删除。一旦定义好查询，就可以再次使用mysql_query()函数来执行该查询，这和其他查询一样。

要检查DELETE查询的执行效果，可以使用mysql_affected_rows()函数。该函数用于返回被INSERT、DELETE或UPDATE查询影响到的行数。

作为示例，我们将编写delete_entry.php脚本，view_blog.php页面中的链接会指向该页面。该页面会接收URL中的数据库记录id。然后它会显示这个条目，确认用户想要删除它（参见图12-22）。如果用户单击了按钮，该记录将被删除（参见图12-23）。

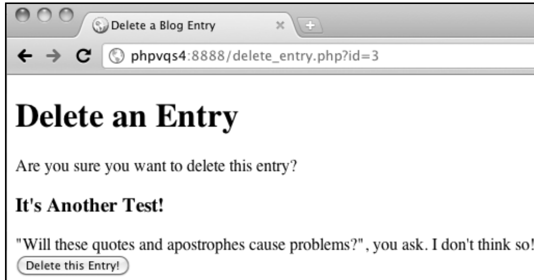


图12-22　当用户进入该页面时，会显示出博客条目，用户必须确认他们想删除它

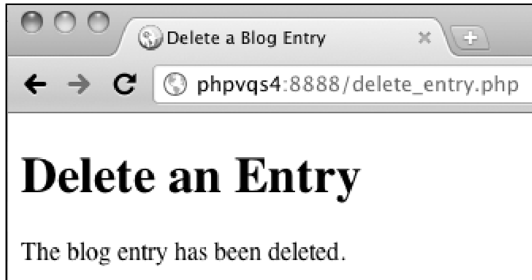


图12-23　如果DELETE查询正确执行了，用户会看到这样的结果

⇛ 从数据库中删除数据

(1) 在文本编辑器或IDE中打开一个新的PHP文档，命名为delete_entry.php（参看脚本12-8）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Delete a Blog Entry </title>
</head>
<body>
<h1>Delete an Entry</h1>
```

脚本12-8　该DELETE SQL命令从表中永久地删除一条记录（或多条记录）

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Delete a Blog Entry</title>
7   </head>
8   <body>
9   <h1>Delete an Entry</h1>
10  <?php // 脚本12-8 - delete_entry.php
11  /* 脚本删除一篇博客。*/
12
13  // 连接服务器并选中数据库:
14  $dbc = mysql_connect('localhost', 'username', 'password');
15  mysql_select_db('myblog', $dbc);
16
```

```
17 if (isset($_GET['id']) && is_numeric($_GET['id']) ) { // 在表单中显示条目：
18
19   // 定义查询：
20   $query = "SELECT title, entry FROM entries WHERE entry_id={$_GET['id']}";
21   if ($r = mysql_query($query, $dbc)) { // 运行查询。
22
23      $row = mysql_fetch_array($r); // 返回信息。
24
25      // 创建表单：
26      print '<form action="delete_entry.php" method="post">
27      <p>Are you sure you want to delete this entry?</p>
28      <p><h3>' . $row['title'] . '</h3>' .
29      $row['entry'] . '<br/>
30      <input type="hidden" name="id" value="' . $_GET['id'] . '" />
31      <input type="submit" name="submit" value="Delete this Entry!" /></p>
32      </form>';
33
34   } else { // 无法获取信息。
35      print '<p style="color: red;">Could not retrieve the blog entry because:<br/>' .
        mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
36   }
37
38 } elseif (isset($_POST['id']) && is_numeric($_POST['id'])) { // 处理表单。
39
40   // 定义查询：
41   $query = "DELETE FROM entries WHERE entry_id={$_POST['id']} LIMIT 1";
42   $r = mysql_query($query, $dbc); // Execute the query.
43
44   // 检查查询结果：
45   if (mysql_affected_rows($dbc) == 1) {
46      print '<p>The blog entry has been deleted.</p>';
47   } else {
48      print '<p style="color: red;">Could not delete the blog entry because:<br/>' .
        mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
49   }
50
51 } else { // 没有获取id。
52    print '<p style="color: red;">This page has been accessed in error.</p>';
53 } // 结束主条件语句。
54
55 mysql_close($dbc); // 关闭连接。
56
57 ?>
58 </body>
59 </html>
```

(2) 打开PHP代码并连接到数据库：

```
<?php // 脚本12-8 - delete_entry.php
$dbc = mysql_connect('localhost', 'username', 'password');
mysql_select_db('myblog', $dbc);
```

(3) 如果页面从URL中接收到了有效的条目`id`，则定义并执行一个`SELECT`查询：

```
if (isset($_GET['id']) && is_numeric($_GET['id']) ) {
```

```
  $query = "SELECT title, entry FROM entries WHERE entry_id={$_GET['id']}";
  if ($r = mysql_query($query, $dbc)) {
```

要显示blog条目，页面必须确认从URL中接收到了一个数值id。由于该id来自于URL（当用户单击view_blog.php中的链接时，参见图12-24），因此通过$_GET['id']进行引用。



图12-24　如果脚本没有接收到ORL中的id值，会打印此错误信息

这个查询和前面例子中使用的SELECT查询类似，区别在于这里添加了WHERE子句，只获取一条特定的记录。另外，由于只有title和entry这两个存储值是必需的，所以这里只选中了这两列。

然后使用mysql_query()函数在数据库上运行该查询。

(4) 检索记录，并在表单中显示条目：

```
$row = mysql_fetch_array($r);
print '<form action="delete_entry.php" method="post">
<p>Are you sure you want to delete this entry?</p>
<p><h3>' . $row['title'] . '</h3>' .
$row['entry'] . '<br/>
<input type="hidden" name="id" value="' . $_GET['id'] . '" />
<input type="submit" name="submit" value="Delete this Entry!" /></p>
</form>';
```

和前面例子中检索所有记录并通过while循环进行处理不同，这里只调用了一次mysql_fetch_array()函数，将返回的记录赋给$row变量。使用该数组可以显示将要被删除的记录。

该表单首先显示博客条目，这和view_blog.php脚本所做的非常类似。当用户单击了按钮，表单会被提交回该页面，此时记录将被删除。完成这一功能，通过$_GET['id']传递到脚本中的blog条目的id值，必须存放到一个隐藏输入框中，这样在提交时它会存在于$_POST数组中（因为此时$_GET['id']中不再有值）。

(5) 如果查询失败，则报告错误：

```
} else { // 无法获取信息。
  print '<p style="color: red;"> Could not retrieve the blog entry because:<br/>' .
  →mysql_error($dbc) . '.</p> <p>The query being run was: ' . $query . '</p>';
}
```

如果SELECT查询运行失败，则MySQL错误和查询本身会被打印出来。

(6) 检查表单是否正在提交：

```
} elseif (isset($_POST['id']) && is_numeric($_POST['id'])) { // 处理表单。
```

这个elseif子句是第3步中开始创建的条件语句的一部分。它对应着该脚本的第二个功能（表单正在提交）。如果该条件是TRUE，则应该删除记录。

(7) 定义和执行查询：

```
$query = "DELETE FROM entries WHERE entry_id={$_POST['id']} LIMIT 1";
$r = mysql_query($query, $dbc);
```

该查询删除entry_id与$_POST['id']值相等的记录。

这里使用的ID值来自于表单，存放在其中的隐藏输入框中。通过向查询添加LIMIT 1子句，可以确保最多只删除一条记录。

(8) 检查查询的结果：

```
if (mysql_affected_rows($dbc) == 1) {
  print '<p>The blog entry has been deleted.</p>';
} else {
  print '<p style="color: red;"> Could not delete the blog entry because:<br/>' .
  →mysql_error($dbc) . '.</p> <p>The query being run was: ' . $query . '</p>';
}
```

mysql_affected_rows()函数返回最近一条查询所修改的行数。如果查询执行成功了，会有一行被删除，所以该函数应该返回1。如果是这样的话，则打印一条消息。否则，出于调试的目的，会打印出MySQL错误和查询本身。

(9) 完成主条件语句：

```
} else { // 没有获取id。
  print '<p style="color: red;"> This page has been accessed in error.</p>';
} // 结束主条件语句。
```

如果既没有使用GET方法也没有使用POST方法将数值id值传递到页面中，则这个else会发生作用（参见图12-24）。

(10) 关闭数据库连接，并完成页面：

```
mysql_close($dbc);
?>
</body>
</html>
```

(11) 将脚本保存为delete_entry.php，放置在启用了PHP的服务器上的适当目录中，并在Web浏览器中测试（参见图12-22和图12-23）。

要测试该脚本，必须首先运行view_blog.php。然后单击其中的Delete链接访问delete_entry.php。

✓提示

- 通过运行TRUNCATE TABLE *tablename*查询可以清空一个表。这种方式要优于使用DELETE FROM *tablename*。TRUNCATE会完全删除并重建一个表，这对数据库来说更好。

- ❑ 试图运行DELETE * FROM *tablename*这样的查询（像编写SELECT查询那样）是一种非常常见的错误。要记得，DELETE和SELECT使用的语法是不同的，因为你不能删除指定的列。

12.10　更新数据库中的数据

本章将要介绍的最后一类查询是UPDATE。它用于修改一条记录中某些列上的值。其语法如下所示：

```
UPDATE tablename SET column1_name=value, column2_name=value2 WHERE some_column=value
```

和任何其他查询一样，如果值是字符串，则应该将其放置在单引号中：

```
UPDATE users SET first_name='Eleanor', age=7 WHERE user_id=142
```

和DELETE查询一样，应该使用一个WHERE子句来限制所影响的行。如果不这么做，则数据库中的每条记录都将被更新。

测试更新的结果，请再次使用mysql_affected_rows()函数，返回修改的记录数量。

作为演示，下面我们编写一个页面用于编辑博客条目。该页面允许用户修改条目的标题和文字，但不能修改输入日期和博客id值（作为主键，id值永远不应改变）。这个脚本使用的结构和delete_entry.php（参见脚本12-8）类似，首先显示条目（参见图12-25），然后处理表单的提交（参见图12-26）。



图12-25　当用户抵达编辑页时，在表单中显示现有的值

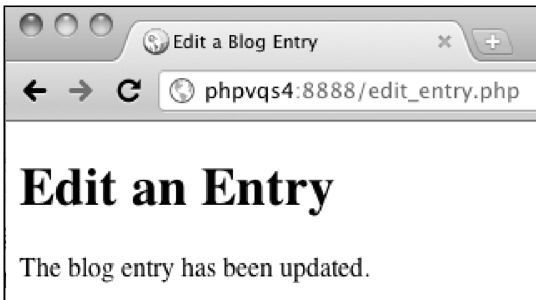


图12-26　继续提交表单，用户会看到这样的消息

⇒ 更新数据库中的数据

(1) 在文本编辑器或IDE中打开一个新的PHP文档，命名为edit_entry.php（参看脚本12-9）。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Edit a Blog Entry</title>
</head>
<body>
<h1>Edit an Entry</h1>
```

脚本12-9 使用UPDATE SQL命令可以编辑数据库表中的记录

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6      <title>Edit a Blog Entry</title>
7   </head>
8   <body>
9   <h1>Edit an Entry</h1>
10  <?php // 脚本12-9 - edit_entry.php
11  /* 脚本使用UPDATE编辑博客条目。*/
12
13  // 连接服务器并选中数据库:
14  $dbc = mysql_connect('localhost', 'username', 'password');
15  mysql_select_db('myblog', $dbc);
16
17  if (isset($_GET['id']) && is_numeric($_GET['id']) ) { // 在表单中显示条目:
18
19     // 定义查询。
20     $query = "SELECT title, entry FROM entries WHERE entry_id={$_GET['id']}";
21     if ($r = mysql_query($query, $dbc)) { // 运行查询。
22
23        $row = mysql_fetch_array($r); // 返回信息。
24
25        // 创建表单:
26        print '<form action="edit_entry.php" method="post">
27    <p>Entry Title: <input type="text" name="title" size="40" maxsize="100"
      value="' . htmlentities($row['title']) . '" /></p>
28    <p>Entry Text: <textarea name="entry" cols="40" rows="5">' . htmlentities
      ($row['entry']) . '</textarea></p>
29    <input type="hidden" name="id" value="' . $_GET['id'] . '" />
30    <input type="submit" name="submit" value="Update this Entry!" />
31    </form>';
32
33    } else { // 无法获取信息。
34       print '<p style="color: red;">Could not retrieve the blog entry because:
         <br/>' . mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
35    }
36
37  } elseif (isset($_POST['id']) && is_numeric($_POST['id'])) { // 处理表单。
38
39    // 验证表单数据并确保安全:
40    $problem = FALSE;
```

12

```
41    if (!empty($_POST['title']) && !empty($_POST['entry'])) {
42       $title = mysql_real_escape_string(trim(strip_tags($_POST['title'])), $dbc);
43       $entry = mysql_real_escape_string(trim(strip_tags($_POST['entry'])), $dbc);
44    } else {
45       print '<p style="color: red;">Please submit both a title and an entry.</p>';
46       $problem = TRUE;
47    }
48
49    if (!$problem) {
50
51       // 定义查询。
52       $query = "UPDATE entries SET title='$title', entry='$entry' WHERE entry_id=
         {$_POST['id']}";
53       $r = mysql_query($query, $dbc); // 执行查询。
54
55       // 打印结果:
56       if (mysql_affected_rows($dbc) == 1) {
57          print '<p>The blog entry has been updated.</p>';
58       } else {
59          print '<p style="color: red;">Could not update the entry because:<br/>' .
            mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
60       }
61
62     } // 一切正常!
63
64 } else { // 没有获取id。
65    print '<p style="color: red;">This page has been accessed in error.</p>';
66 } // End of main IF.
67
68 mysql_close($dbc); // 关闭连接。
69
70 ?>
71 </body>
72 </html>
```

(2) 打开PHP代码节并连接到数据库：

```
<?php // 脚本12-9 - edit_entry.php
$dbc = mysql_connect('localhost', 'username', 'password');
mysql_select_db('myblog', $dbc);
```

(3) 如果页面从URL中接收到了有效的条目id，则定义并执行一个SELECT查询：

```
if (isset($_GET['id']) && is_numeric($_GET['id']) ) {
  $query = "SELECT title, entry FROM entries WHERE entry_id={$_GET['id']}";
  if ($r = mysql_query($query, $dbc)) {
```

这段代码和删除页中的代码完全一样，针对提供的id值从数据库中检索两列值。

(4) 检索记录，在一个表单中显示条目：

```
$row = mysql_fetch_array($r);
  print '<form action="edit_entry.php" method="post">
<p>Entry Title: <input type="text" name="title" size="40" maxsize="100" value="' .
→htmlentities($row['title']) . '" /></p>
<p>Entry Text: <textarea name="entry" cols="40" rows="5">' .
```

```
→htmlentities($row['entry']) . '</textarea></p>
<input type="hidden" name="id" value="' . $_GET['id'] . '" />
<input type="submit" name="submit" value="Update this Entry!" />
</form>';
```

这段代码也是和前面的脚本完全一样，包括最重要的、在表单的隐藏输入框中存放id值这一步。然而，这里并不是仅仅打印出存储的数据，而是将其用作表单元素的值。为了安全和避免潜在的冲突，每个值在使用之前都用htmlentities()先进行了处理。

(5) 如果查询失败则报告一个错误：

```
} else { // 无法获取信息。
  print '<p style="color: red;">Could not retrieve the blog entry because:
  →<br/>' . mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
}
```

(6) 检查表单的提交：

```
} elseif (isset($_POST['id']) && is_numeric($_POST['id'])) {
```

当表单正在提交时，该条件将为TRUE。

(7) 验证表单数据并确保安全：

```
$problem = FALSE;
if (!empty($_POST['title']) && !empty($_POST['entry'])) {
  $title = mysql_real_escape_string(trim(strip_tags($_POST ['title'])), $dbc);
  $entry = mysql_real_escape_string(trim(strip_tags($_POST ['entry'])), $dbc);
} else {
  print '<p style="color: red;"> Please submit both a title and an entry.</p>';
  $problem = TRUE;
}
```

这段代码来自于用于添加博客条目的页面。它对提交的数据进行了最小的验证，然后在其上运行了mysql_real_escape_string()函数使其变得安全。因为表单数据是可以编辑的，所以应该像对待创建新记录那样对其进行验证。

(8) 定义和执行查询：

```
if (!$problem) {
  $query = "UPDATE entries SET title='$title', entry='$entry'
  →WHERE entry_id={$_POST['id']}";
  $r = mysql_query($query, $dbc);
```

UPDATE查询将title列的值设置为在表单中名为title的输入框控件里输入的值，并将entry列的值设置为在表单中名为entry的textarea控件中输入的值。只有当记录的entry_id等于$_POST['id']，这个来自于表单中的隐藏输入框时，它才会被更新。

(9) 报告查询的成功执行：

```
if (mysql_affected_rows($dbc) == 1) {
  print '<p>The blog entry has been updated.</p>';
} else {
  print '<p style="color: red;"> Could not update the entry because:<br/>' . mysql_
  →error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
}
```

如果恰好有一行受到影响，则会返回一条成功的消息。否则，MySQL错误和查询自身会被发送到Web浏览器。

(10) 完成条件语句：

```
  } // 一切正常!
} else { // 没有获取id。
  print '<p style="color: red;"> This page has been accessed in error.</p>';
} // End of main IF.
```

如果既没有通过GET方法也没有通过POST方法将数字id值传递给该页面，则这个else子句会发生作用。

(11) 关闭数据库连接，并完成页面：

```
mysql_close($dbc);
?>
</body>
</html>
```

(12) 将文件保存为edit_entry.php，放置在启用了PHP的服务器上的适当目录中，并在Web浏览器中进行测试（参见图12-25和图12-26）。

和前面的示例一样，要编辑一个条目，需要首先访问view_blog.php页面并单击其中的Edit链接。

(13) 再次访问view_blog.php，确认已经发生了更改（参见图12-27）。

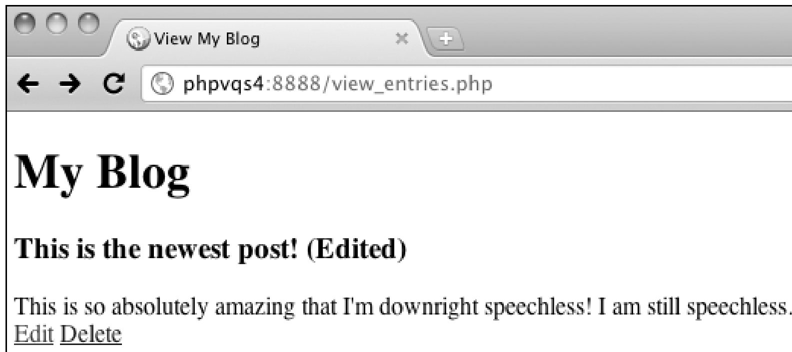


图12-27　重新加载view_blog.php脚本，查看对条目作出的修改

✓提示

- id是一个主键，这意味着它的值永远不应改变。通过在表中使用主键，可以修改某一行记录中所有其他列上的值，但仍然能够用主键列来引用这一行记录。
- 无须在ID值上应用mysql_real_escape_string()函数，因为is_numeric()测试已经确保了它不可能包含撇号或其他有问题的字符。
- 编辑和删除页中多次在条件语句中使用了mysql_num_rows()，在访问SELECT查询的结果之前确保其中有一行数据：

```
if (mysql_num_rows($r) = = 1) {...
```

- 如果在表中执行了一个更新，但没有修改任何记录的值，则mysql_affected_rows()将会返回0。
- 不要忘记向UPDATE查询添加一个LIMIT 1子句，确保最多只影响到一行。

12.11 回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

12.11.1 回顾

- 你使用的是哪个版本的MySQL？你使用的连接MySQL的语句是什么？
- PHP脚本是如何与MySQL服务器连接的？如何断开连接？
- 什么是错误抑制运算符，作用是什么？
- 哪个函数可以返回MySQL报告的错误？
- 在使用PHP脚本与MySQL交互时，如果出现问题，应该使用什么调试技术？
- 如何选择数据库？
- 创建一张表的SQL命令是什么？添加新记录、检索记录、修改记录、删除记录的命令都是什么？
- 字符串值应该使用什么函数来防止SQL注入攻击？

12.11.2 实践

- 阅读与你的MySQL版本相同的MySQL用户手册。
- 将连接和选择数据库的代码保存到一个单独的脚本中，然后在与数据库交互的PHP页面中引入这个脚本。
- 将add_entry.php表单改为粘性表单。
- 修改view_entry.php中的代码，将输入的所有换行符转换成HTML的换行标签。

第 13 章 将所有的组合在一起

13

本章内容

- 准备开始
- 连接数据库
- 编写用户自定义函数
- 创建模板
- 登录
- 登出
- 添加名人名言
- 显示名人名言
- 编辑名人名言
- 删除名人名言
- 创建主页
- 回顾和实践

前几章介绍了PHP用于Web开发的所有基础知识。这一章，我们会运用所有学过的知识创建一个功能完备的网站。当然，这一章也会介绍一些新的知识，你会从中掌握很多新的技巧。特别是你会看到如何从零开始开发一个完整的Web应用程序。

13.1 准备开始

在开始任何网站项目之前，都要先制定目标。这一章的主要目标是应用前面学过的所有知识（绝对是一个崇高的目标）。这一章的例子会将前两章使用的模块组合在一起。我们会创建一个网站，该网站可以存储和显示名人名言。本章不会像第11章那样使用文件存储名人名言，而是用MySQL数据库存储用户提交的名人名言。就像第12章的博客示例一样，这一章也会实现创建、编辑和删除名人名言。默认情况下，公共用户可以在主页上看到最新添加的名言（参见图13-1），主页上也可以显示随机名言或随机的受欢迎的名言。

为了提高网站安全性，这个网站会有一个管理员，可以登录和登出网站。只有以管理员身份

登录后，才可以创建、编辑和删除名言（参见图13-2）。



图13-1 网站简单的主页



图13-2 非管理员会被拒绝访问某些页面

这个网站会使用一个简单的模板，以保证所有页面的显示效果一致。我们会使用CSS处理所有版式和布局。网站还会使用一个用户定义函数，我们将这个函数写在一个包含文件中。

和第12章一样，我们会首先创建数据库和一个表。数据库可以命名为myquotes（你也可以自己起个名字）。用以下SQL命令创建数据表：

```
CREATE TABLE quotes (
  quote_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  quote TEXT NOT NULL,
  source VARCHAR(100) NOT NULL,
  favorite TINYINT(1) UNSIGNED NOT NULL,
  date_entered TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (quote_id)
)
```

quote_id是主键，每当添加一个新的名言时，这个值都会递增。quote字段用于存储名言，source字段用于存储名言出处，即人名（不同于第11章将名言和人名一起存储）。favorite字段存储1或0，用于标记名言是否受欢迎。date_entered字段是一个时间戳，当创建一个新记录时会自动设置当前时间戳。

可以使用PHP脚本创建这个表（参看脚本12-4），也可以使用第三方应用程序（例如，MySQL客户端或phpMyAdmin）。

最后介绍一下服务器中网站的文件组织结构（参见图13-3）。具体地说，用于创建数据库连接的mysql_connect.php脚本应该存储在Web根目录的外部，如果条件不允许这样做，可以把它放在includes文件夹下面，注意修改其他相关脚本中的代码。

创建完数据库、数据库表和必要的文件夹之后，就可以开始编码工作了。

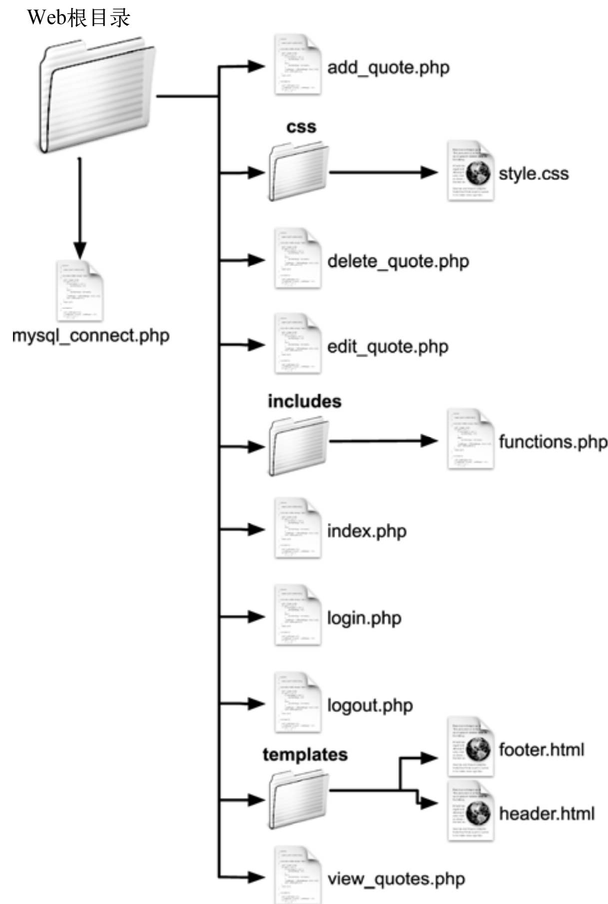


图13-3　服务器上的文件组织结构

13.2　连接数据库

在第12章中，为了连接数据库，我们在所有脚本中添加了同样的代码。这回我们使用更常用的方法，就是将那些需要重复使用的代码放到一个独立的文件中。每个与数据库交互的脚本（大部分脚本）都会包含这个文件。如图13-3所示，数据库连接文件应当放在Web根目录的外面，如果条件不允许你这样做，你可以把它放在`includes`文件夹下面。

⇒ 创建mysql_connect.php

(1) 在文本编辑器或IDE中新建一个脚本，命名为`mysql_connect.php`（参看脚本13-1）：

```
<?php
```

脚本13-1　这个脚本连接数据库服务器，并选择使用的数据库

```
1   <?php // 脚本13-1 - mysql_connect.php
2   /* 脚本连接服务器并选中数据库。*/
3
4   // 连接：
5   $dbc = mysql_connect('localhost', 'username', 'password');
6
7   // 选中：
8   mysql_select_db('myquotes', $dbc);
9
10  ?>
```

(2) 连接数据库服务器：

```
$dbc = mysql_connect('localhost','username', 'password');
```

你可以根据自己使用的服务器情况改变这些值。

(3) 选择数据库：

```
mysql_select_db('myquotes', $dbc);
```

你可以根据要连接的数据库名称，修改相应的值。

(4) 结束脚本：

```
?>
```

(5) 将脚本保存为mysql_connect.php。

13.3　编写用户定义函数

这个网站包含一个用户定义函数。如第10章所述，如果脚本或网站中有很多重复的代码，最好创建自定义函数。这个网站有很多这种情况，最明显的例子是：很多脚本都需要检查当前用户是否为管理员。下面的脚本会创建用户定义函数，它返回一个布尔值，表示用户是否为管理员。但是脚本如何测试是不是管理员呢？

如果成功登录网站，服务器会将一个名为Samuel、值为Clemens的cookie（参见图13-4）发送到管理员的浏览器。这可能看起来很奇怪或随机，但事实就是如此。当你使用简单的方式（比如cookie）做验证时，最好使用这种不太明显的验证组合。如果用很清楚的验证组合，比如名为admin值为TRUE，那么任何人都可以猜到它并伪造它。了解了这些之后，以下的函数只会检查是否存在名为Samuel、值为Clemens的cookie。

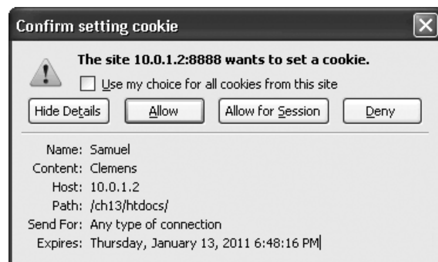


图13-4　这个cookie用于辨认管理员

⇛ 创建`functions.php`

(1) 在文本编辑器或IDE中新建一个脚本，命名为`functions.php`（参看脚本13-2）：

```
<?php // 脚本13-2 - functions.php
```

虽然这个脚本只定义了一个自定义函数，但却用了复数形式（functions）命名，因为将来可能需要向这个脚本中添加其他用户自定义函数。

脚本13-2　`is_administrator()`函数定义在了一个可包含脚本中，每个需要验证管理员状态的页面都可以调用此脚本

```
1   <?php // 脚本13-2 - functions.php
2   /* 脚本定义自定义函数。*/
3   
4   // 函数检查用户是否为管理员。
5   // 函数接受两个可选参数。
6   // 函数返回一个布尔值。
7   function is_administrator($name = 'Samuel', $value = 'Clemens') {
8   
9     // 检查cookie是否存在和cookie值：
10    if (isset($_COOKIE[$name]) && ($_COOKIE[$name] == $value)) {
11        return true;
12    } else {
13        return false;
14    }
15  
16  } // 结束is_administrator()函数。
17  
18  ?>
```

(2) 定义一个新函数：

```
function is_administrator($name = 'Samuel', $value = 'Clemens') {
```

函数接受两个参数：cookie的名和值。两个参数都有默认值。

由于这个函数只检查某个特定值的cookie，所以其实并没有必要设置参数及其默认值。设置这些参数是出于将来扩展网站功能的目的（比如，如果你要执行多种类型的验证）。使用带默认值的参数，可以在调用函数时省去参数值，前提是已经验证过它们。

(3) 根据cookie是否存在和cookie的值，返回布尔值：

```
if (isset($_COOKIE[$name]) && ($_COOKIE[$name] == $value)) {
    return true;
} else {
    return false;
}
```

如果cookie存在，并且cookie值正确，会返回TRUE，否则函数返回FALSE。

(4) 结束函数和脚本：

```
} // 结束is_administrator()函数。
?>
```

(5) 将脚本保存为functions.php，存放在includes目录下。

13.4　创建模板

我们已经创建了两个辅助文件，接下来创建模板。如第8章所述，网站的布局由两个可包含文件控制：头文件和页脚文件。这两个文件会存放在templates目录中。头文件会引用一个样式表，样式表文件存放在css文件夹下。你可以在本书第13章的源代码（ch13文件夹下）中找到这个样式表。（源代码可以从www.LarryUllman.com下载。）

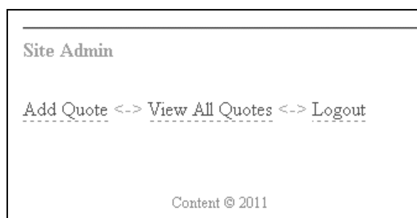


图13-5　如果访问页面的用户是管理员，他会看到一些管理链接

除了产生网页主要的HTML之外，头文件还包含自定义函数脚本。如果当前用户是管理员，页脚文件还会显示一些管理链接（参见图13-5）。

⇛ 创建头文件

(1) 在文本编辑器或IDE中新建一个HTML文档，命名为header.html（参看脚本13-3）：

```
<?php // 脚本13-3 - header.html
include('includes/functions.php'); ?>
```

头文件以PHP代码开头，目的是包含functions.php脚本，网站中的很多页面都会用到此脚本。使用相对路径引用被包含脚本，因为包含头文件的页面都存放在主目录中。

脚本13-3　头文件包含functions.php脚本，并创建HTML页面

```
1   <?php // 脚本13-3 - header.html
2
3   // 包含functions脚本:
4   include('includes/functions.php'); ?>
5   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
6     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
7   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
8   <head>
9      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
10     <link rel="stylesheet" media="all" href="css/style.css" />
11     <title><?php // 打印页面标题。
12     if (defined('TITLE')) { // Is the title defined?
13          print TITLE;
14     } else { // 标题未定义。
15       print 'My Site of Quotes';
16     }
17     ?></title>
18  </head>
19  <body>
20    <div id="container">
21      <h1>My Site of Quotes</h1>
22      <br/>
```

```
23      <!-- 可变内容开始。-->
```

(2) 开始编写HTML文档：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <link rel="stylesheet" media="all" href="css/style.css" />
```

可以从本书的源代码中找到这个样式表。使用相对路径引用样式表，因为包含头文件的页面都存放在主目录中。

(3) 打印页面标题：

```
<title><?php
if (defined('TITLE')) {
  print TITLE;
} else {
  print 'My Site of Quotes';
}
?></title>
```

这段代码和第8章类似，根据是否定义了title常量，打印默认标题或自定义标题。

(4) 结束head标签：

```
</head>
```

(5) 开始页面body部分：

```
<body>
  <div id="container">
    <h1>My Site of Quotes</h1>
    <br/>
    <!-- BEGIN CHANGEABLE CONTENT. -->
```

(6) 将脚本保存为header.html，存放在templates目录下。

⇛ 创建footer.html

(1) 在文本编辑器或IDE中新建一个HTML文档，命名为footer.html（参看脚本13-4）：

```
<!-- 可变内容结束。-->
```

脚本13-4　页脚文件在恰当的时候显示常用管理链接，同时完成HTML页面

```
1   <!-- END CHANGEABLE CONTENT. -->
2   <?php // 脚本13-4 - footer.html
3
4   // 显示管理链接……
5   // - 如果用户是管理员且当前页面不为logout.php
6   // - 或者$loggedin变量为TRUE（例如，用户刚刚登录）
7   if ( (is_administrator() && (basename($_SERVER['PHP_SELF']) != 'logout.php'))
8   OR (isset($loggedin) && $loggedin) ) {
9
```

```
10      // 创建链接:
11      print '<hr /><h3>Site Admin</h3><p><a href="add_quote.php">Add Quote</a> <->
12      <a href="view_quotes.php">View All Quotes</a> <->
13      <a href="logout.php">Logout</a></p>';
14
15  }
16
17  ?>
18      </div><!-- 容器 -->
19      <div id="footer">Content &copy; 2011</div>
20  </body>
21  </html>
```

(2) 检查是否需要显示常用管理链接：

```
<?php
if ( (is_administrator() && (basename($_SERVER['PHP_SELF'])!= 'logout.php'))
OR (isset($loggedin) && $loggedin) ) {
```

这个条件语句有些复杂。虽然很多页面都可以通过调用`is_administrator()`函数确认当前用户是否为管理员，但是由于cookie的工作方式，在以下两个页面中这个函数会返回不正确的结果：`login.php`和`logout.php`。在`logout.php`页面上，脚本会在收到管理员cookie之后删除它。也就是说，管理员在该页面中会看不到管理链接（因为当他们访问这个页面时cookie已经不存在了）。条件语句的前面部分需要`is_administrator()`函数返回TRUE并且当前页不能是`logout.php`。`basename($_SERVER['PHP_SELF'])`代码是获取当前脚本的可靠方法。（由于页脚文件包含在另一个脚本中，`$_SERVER['PHP_SELF']`也有包含脚本的值。）

条件语句的后半部分（`OR`之后），检查是否设置了`$loggedin`且其值是否为TRUE。这是为了应对当用户成功登录`login.php`页面时的cookie问题。`is_administrator()`函数在用户刚刚登录后不会返回TRUE，因为这时cookie刚刚通过脚本发送到客户端，所以不可能被读取到。

(3) 创建链接：

```
print '<hr /><h3>Site Admin</h3><p><a href="add_quote.php">Add
→Quote</a> <->
<a href="view_quotes.php">View All Quotes</a> <->
<a href="logout.php">Logout</a></p>';
```

创建了3个链接，Add Quote（用于添加新名言）、View All Quotes（查看所有名言）和Logout（登出）。

(4) 完成条件语句和PHP代码：

```
}
?>
```

(5) 完成HTML页面：

```
  </div><!-- 容器 -->
  <div id="footer">Content &copy;2011</div>
  </body>
  </html>
```

(6) 将文件保存为footer.html，并存放在templates目录下。

13.5　登录

接下来，创建管理员登录脚本。最终结果会和第9章的脚本类似，只是有一个结构上的区别：第9章使用的是输出缓存，允许任意安排脚本。本章的示例不使用输出缓存，因此在处理表单时，必须保证发送cookie而不导致headers already sent erros错误（参见第9章）。换句话说，脚本在包含头文件之前会先检查表单提交情况。为了仍然能够捕捉到页面中的错误信息和其他信息，你必须使用几个变量。

⇒ 创建头文件

(1) 在文本编辑器或IDE中新建一个PHP文档，命名为login.php（参看脚本13-5）：

```
<?php // 脚本13-5 - login.php
```

脚本13-5　登录页面会显示和处理表单，在成功登录后会发送cookie

```
1     <?php // 脚本13-5 - login.php
2     /* 页面用于用户登录网站。*/
3
4     // 使用默认值定义两个变量：
5     $loggedin = false;
6     $error = false;
7
8     // 检查表单是否已提交：
9     if ($_SERVER['REQUEST_METHOD'] == 'POST')
{
10
11      // 处理表单：
12      if (!empty($_POST['email']) && !empty($_POST['password'])) {
13
14        if ( (strtolower($_POST['email']) == 'me@example.com') && ($_POST['password']
          == 'testpass') ) { // 相等!
15
16           // 创建cookie：
17           setcookie('Samuel', 'Clemens', time()+3600);
18
19           // 标识已登录：
20           $loggedin = true;
21
22        } else { // 不相等!
23
24           $error = 'The submitted email address and password do not match those
             on file!';
25
26        }
27
28      } else { // 表单未填完整。
29
30         $error = 'Please make sure you enter both an email address and a password!';
```

```
31
32    }
33
34  }
35
36  // 设置页面标题并包含页头文件：
37  define('TITLE', 'Login');
38  include('templates/header.html');
39
40  // 在出现错误时打印错误信息：
41  if ($error) {
42    print '<p class="error">' . $error . '</p>';
43  }
44
45  // 检查用户是否已登录，如没有登录则显示表单：
46  if ($loggedin) {
47
48    print '<p>You are now logged in!</p>';
49
50  } else {
51
52     print '<h2>Login Form</h2>
53     <form action="login.php" method="post">
54     <p><label>Email Address <input type="text" name="email" /></label></p>
55     <p><label>Password <input type="password" name="password" /></label></p>
56     <p><input type="submit" name="submit" value="Log In!" /></p>
57     </form>';
58
59  }
60
61  include('templates/footer.html'); // 包含页脚文件。
62  ?>
```

(2) 使用默认值定义两个变量：

```
$loggedin = false;
$error = false;
```

脚本接下来会使用这两个变量。这里设置了变量的默认值，表示用户没有登录，也没有发生错误。

(3) 检查表单是否已经提交：

```
if ($_SERVER['REQUEST_METHOD'] =='POST') {
```

(4) 处理表单：

```
if (!empty($_POST['email']) && !empty($_POST['password'])) {
  if ( (strtolower($_POST['email']) == 'me@example.com') && ($_POST
  ➝['password'] == 'testpass') ) {
```

类似第9章的示例，这个脚本首先确认`$_POST['email']`和`$_POST['password']`非空。第二个条件语句检查提交数据是否符合要求。

(5) 创建cookie：

```
setcookie('Samuel', 'Clemens', time()+3600);
```

这个cookie的名为Samuel，值为Clemens。设置了1小时的过期时间。

(6) 显示用户已登录：

```
$loggedin = true;
```

这个脚本的后面和页脚文件中（参看脚本13-4）都会使用这个变量。

(7) 为其他两个条件分支创建错误信息：

```
    } else { // 不相等！
      $error = 'The submitted email address and password do not match those on file!';
    }
  } else { // 表单未填完整。
    $error = 'Please make sure you enter both an email address and a password!';
  }
}
```

第1个else子句处理提交了错误的邮件地址和密码的情况。第2个else子句处理没有提交邮件地址或密码的情况。这两种情况都将消息发送给一个变量，这个变量会在脚本后面的代码中用到。

(8) 设置页面标题并包含头文件：

```
define('TITLE', 'Login');
include('templates/header.html');
```

(9) 在出现错误时，打印错误信息：

```
if ($error) {
  print '<p class="error">' . $error . '</p>';
}
```

错误的类型会在第(7)步时确定，但并没有在第(7)步打印出来，因为那时还没有包含头文件。解决方法是让代码检查非FALSE的$error值，然后在HTML和CSS中打印$error（参见图13-6）。

图13-6　在包含头文件之后，HTML表单之前显示错误信息

(10) 显示用户已登录，否则显示登录表单：

```
if ($loggedin) {
  print '<p>You are now logged in!</p>';
} else {
  print '<h2>Login Form</h2>
  <form action="login.php" method="post">
  <p><label>Email Address <input type="text" name="email" /> </label></p>
  <p><label>Password <input type="password" name="password" /></label></p>
```

```
    <p><input type="submit" name="submit" value="Log In!" /></p>
    </form>';
}
```

如果`$loggedin`变量的值为TRUE（默认值为FALSE），就表明用户已经成功登录网站，这时会显示登录成功的信息（参见图13-7）。如果`$loggedin`变量的值仍为FALSE，就会显示登录表单（参见图13-8）。

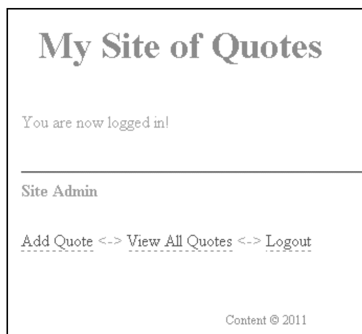


图13-7 登录成功后的页面



图13-8 基本登录表单

(11) 包含页脚文件并完成页面：

```
include('templates/footer.html');
?>
```

(12) 将文件保存为`login.php`。

(13) 在Web浏览器中测试文件（参见图13-6、图13-7和图13-8）。

我故意没有创建连接到登录部分的链接（出于安全方面的考虑），你需要在Web浏览器的地址栏中输入正确的URL。

✓提示

- ❑ 注意，这个脚本需要管理员在1小时之后再次登录，无论他是否继续在网站中活动。

13.6 登出

既然写了登录程序，就应该有登出程序。以下是一个简单的登出脚本，只使用了cookie。

⇒ 创建`logout.php`

(1) 在文本编辑器或IDE中新建一个PHP文档，命名为`logout.php`（参看脚本13-6）：

```
<?php // 脚本13-6 - logout.php
```

脚本13-6 登出脚本删除管理员标识cookie

```
1   <?php // 脚本13-6 - logout.php
2   /* 这是登出页面，它会删除cookie。*/
3
```

```
4   // 如果cookie存在，则删除：
5   if (isset($_COOKIE['Samuel'])) {
6     setcookie('Samuel', FALSE, time()-300);
7   }
8
9   // 定义页面标题并包含页头文件：
10  define('TITLE', 'Logout');
11  include('templates/header.html');
12
13  // 打印一条消息：
14  print '<p>You are now logged out.</p>';
15
16  // 包含页脚文件：
17  include('templates/footer.html');
18  ?>
```

(2) 如果cookie存在，删除它：

```
if (isset($_COOKIE['Samuel'])) {
  setcookie('Samuel', FALSE, time()-300);
}
```

为了安全起见，这个脚本只会在cookie已经存在时才删除它。黑客可以在不登录的情况下访问登出脚本来获取cookie名，而以上查询语句则可以防止出现这种安全问题。

要删除已经存在的登录cookie，可以发送另一个同名cookie，将值设置为FALSE，并将过期时间设置为一个过去的时间。

(3) 定义页标题并包含头文件：

```
define('TITLE', 'Logout');
include('templates/header.html');
```

(4) 打印登出信息：

```
print '<p>You are now logged out.</p>';
```

(5) 包含页脚文件：

```
include('templates/footer.html');
?>
```

(6) 将文件保存为logout.php。

(7) 在Web浏览器中测试文件（参见图13-9和图13-10）。

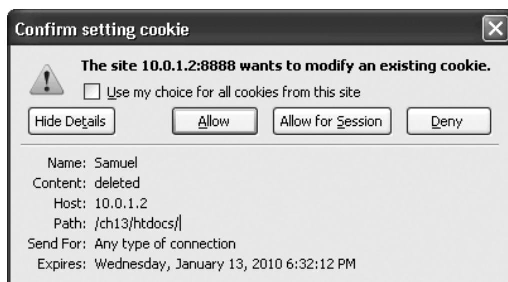


图13-9　这个cookie删除已经存在的登录cookie



图13-10　登出后的页面

13.7　添加名人名言

现在管理员已经可以登录网站了，也可以添加名言了。这个脚本与第12章的添加blog文章的示例类似，只是脚本中的表单会增加一个复选框，用于标记引用语是否是受欢迎的名人名言（参见图13-11）。

图13-11　添加名人名言到数据库的表单

由于这个脚本要向数据库中添加记录，所以要特别注意安全问题。首先，只有管理员才可以使用这个页面。其次，需要验证SQL命令中使用的值，防止无效查询。

⇒ 创建add_quote.php

(1) 在文本编辑器或IDE中新建一个PHP文档，命名为add_quote.php（参看脚本13-7）：

```
<?php // 脚本13-7 - add_quote.php
define('TITLE', 'Add a Quote');
include('templates/header.html');
print '<h2>Add a Quotation</h2>';
```

脚本13-7　add_quote.php脚本只允许管理员添加新的名人名言到数据库中

```
1   <?php // 脚本13-7 - add_quote.php
2   /* 脚本添加名人名言。*/
3
4   // 定义页面标题并包含页头文件。
5   define('TITLE', 'Add a Quote');
6   include('templates/header.html');
7
8   print '<h2>Add a Quotation</h2>';
9
10  // 强制只有管理员可以访问该页面:
11  if (!is_administrator()) {
12     print '<h2>Access Denied!</h2><p class="error">You do not have permission
```

```
        to access this page.</p>';
13      include('templates/footer.html');
14      exit();
15  }
16
17  // 检查表单是否提交：
18  if ($_SERVER['REQUEST_METHOD'] == 'POST') { // 处理表单。
19
20       if ( !empty($_POST['quote']) && !empty($_POST['source']) ) {
21
22       // 包含数据库连接。
23       include('../mysql_connect.php');
24
25       // 准备查询中使用的值：
26       $quote = mysql_real_escape_string(trim(strip_tags($_POST['quote'])), $dbc);
27       $source = mysql_real_escape_string(trim(strip_tags($_POST['source'])), $dbc);
28
29       // 创建favorite值：
30       if (isset($_POST['favorite'])) {
31          $favorite = 1;
32       }  else {
33          $favorite = 0;
34       }
35
36       $query = "INSERT INTO quotes (quote, source, favorite) VALUES ('$quote',
         '$source', $favorite)";
37       $r = mysql_query($query, $dbc);
38
39       if (mysql_affected_rows($dbc) == 1) {
40          // 打印一条消息：
41          print '<p>Your quotation has been stored.</p>';
42       } else {
43          print '<p class="error">Could not store the quote because:<br/>' .
            mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
44       }
45
46       // 关闭连接：
47       mysql_close($dbc);
48
49    } else { // 没有填写名人名言。
50       print '<p class="error">Please enter a quotation and a source!</p>';
51    }
52
53  } // 结束提交条件语句。
54
55  // 结束PHP节并显示表单：
56  ?>
57
58  <form action="add_quote.php" method="post">
59     <p><label>Quote <textarea name="quote" rows="5" cols="30">
       </textarea></label></p>
60     <p><label>Source <input type="text" name="source" /></label></p>
61     <p><label>Is this a favorite? <input type="checkbox" name="favorite" value=
       "yes" /></label></p>
```

```
62     <p><input type="submit" name="submit" value="Add This Quote!" /></p>
63  </form>
64
65  <?php include('templates/header.html'); ?>
```

(2) 如果用户不是管理员，将不能访问这个页面：

```
if (!is_administrator()) {
  print '<h2>Access Denied!</h2><p class="error">You do not have permission to access
  ➝this page.</p>';
  include('templates/footer.html');
  exit();
}
```

通过调用`is_administrator()`函数，脚本可以检查用户是否为管理员。如果用户不是管理员，会显示Access Denied错误信息，后面的代码用于包含页脚文件和结束脚本（参见图13-12）。



图13-12　如果用户不是管理员，将不能访问这个页面

(3) 检查表单提交情况：

```
if ($_SERVER['REQUEST_METHOD'] =='POST') {
```

(4) 检查表单中的值：

```
if ( !empty($_POST['quote']) &&!empty($_POST['source']) ) {
```

这个脚本只进行了最少量的表单验证，检查两个变量是否为空。因为要输入的是一大块文本（名人名言），也没有其他什么需要验证的内容。

(5) 准备查询中使用的值：

```
$quote = mysql_real_escape_string(trim(strip_tags($_POST['quote'])), $dbc);
$source = mysql_real_escape_string(trim(strip_tags($_POST['source'])), $dbc);
```

为了使这两个文本值能够在查询中安全地使用，需要使用`mysql_real_escape_string()`函数（参见第12章）。为了使这些值之后可以在Web浏览器中正常显示，需要使用`strip_tags()`函数（参见第5章）。

(6) 创建`favorite`值：

```
if (isset($_POST['favorite'])) {
  $favorite = 1;
```

```
} else {
  $favorite = 0;
}
```

在数据库中，名人名言的受欢迎状态用1和0表示：1为受欢迎，0为不受欢迎。为了确定使用哪个数字，所有的PHP脚本都会检查$_POST ['favorite']的值。如果设置了这个变量，无论值为什么，都表示用户选中了Is this a favorite?复选框。如果用户没有选中它，就不会设置这个变量，那就会将0赋给$favorite变量。

(7) 定义并执行查询：

```
$query = "INSERT INTO quotes (quote, source, favorite) VALUES ('$quote',
  →'$source', $favorite)";
$r = mysql_query($query, $dbc);
```

这个查询使用已经定义的变量分别为3个字段指定了值。程序会自动给其他两个字段（quote_id和date_entered）赋值，这是在定义表的时候设置的。

(8) 根据结果打印相应信息：

```
if (mysql_affected_rows($dbc) == 1) {
  print '<p>Your quotation has been stored.</p>';
} else {
  print '<p class="error">Could not store the quote because:<br/>' .
  →mysql_error($dbc) .'.</p><p>The query being run was: ' . $query . '</p>';
}
```

如果INSERT查询在数据库中添加了一个新行，就会显示“Your quotation has been stored.”信息。否则，就会显示调试信息，你可以从中查找错误原因。

(9) 完成验证条件语句：

```
} else { // 没有填写名人名言
  print '<p class="error">Please enter a quotation and a source!</p>';
}
```

(10) 完成提交条件语句和PHP代码：

```
} // 结束提交条件语句。
?>
```

(11) 创建表单：

```
<form action="add_quote.php" method="post">
  <p><label>Quote <textarea name="quote" rows="5" cols="30"></textarea>
  →</label></p>
  <p><label>Source <input type="text" name="source" /> </label></p>
  <p><label>Is this a favorite? <input type="checkbox" name="favorite" /></label></p>
  <p><input type="submit" name="submit" value="Add This Quote!" /></p>
</form>
```

这个表单由一个文本区、一个文本输入框和一个复选框（当然还包含提交按钮）组成。表单现在不是一个粘性表单，你可以自己加入这个特性。

(12) 包含页脚文件：

```
<?php include('templates/footer.html'); ?>
```

(13) 将文件保存为add_quote.php，并在Web浏览器中测试（参见图13-13）。



图13-13　添加名人名言后的页面

✓提示

❑ 新建（INSERT）、检索（SELECT）、更新和删除数据库记录的功能简写为CRUD（Create、Retrieve、Update、Delete）。

13.8　显示名人名言

网站的管理部分需要有一个页面能够列出所有存储在数据库中的名人名言（参见图13-14）。虽然这个脚本也可以用于公开的页面，但它的主要目的是为管理员提供一些快捷链接，以便编辑、删除那些名言。（这不同于在公开页面随机搜索某个名言来进行管理。）

与add_quote.php脚本类似，这个页面也只允许管理员访问。

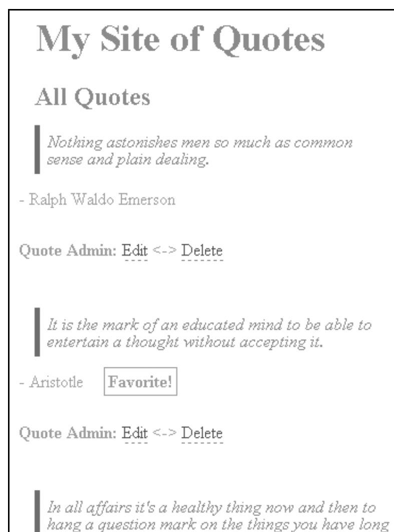


图13-14　所有名人名言的列表，每个名人名言下都有编辑和删除链接

⇒ 创建view_quotes.php

(1) 在文本编辑器或IDE中新建一个PHP文档，命名为view_quotes.php（参看脚本13-8）：

```
<?php // 脚本13-8 - view_quotes.php
define('TITLE', 'View All Quotes');
include('templates/header.html');
print '<h2>All Quotes</h2>';
```

脚本13-8　这个脚本显示当前存储的所有名人名言，并为管理员提供了编辑和删除链接

```
1   <?php // 脚本13-8 - view_quotes.php
2   /* 脚本显示所有名人名言。*/
3
4   // 包含页头文件：
5   define('TITLE', 'View All Quotes');
6   include('templates/header.html');
7
8   print '<h2>All Quotes</h2>';
9
10  // 强制只有管理员可以访问该页面：
11  if (!is_administrator()) {
12     print '<h2>Access Denied!</h2><p class="error">You do not have permission to
       access this page.</p>';
13     include('templates/footer.html');
14     exit();
15  }
16
17  // 包含数据库连接：
18  include('../mysql_connect.php');
19
20  // 定义查询：
21  $query = 'SELECT quote_id, quote, source, favorite FROM quotes ORDER BY date_entered DESC';
22
23  // 运行查询：
24  if ($r = mysql_query($query, $dbc)) {
25
26      // 返回查询结果：
27    while ($row = mysql_fetch_array($r)) {
28
29      // 打印查询结果：
30      print "<div><blockquote>{$row['quote']}</blockquote>- {$row['source']}\n";
31
32      // 判断是否为受欢迎的。
33      if ($row['favorite'] == 1) {
34          print ' <strong>Favorite!</strong>';
35      }
36
37      // 添加管理员链接：
38      print "<p><b>Quote Admin:</b> <a href=\"edit_quote.php?id={$row['quote_id']}\">
        Edit</a> <->
39      <a href=\"delete_quote.php?id={$row['quote_id']}\">Delete</a></p></div>\n";
40
41    } // 结束循环。
```

```
42
43  } else { // 没有运行查询。
44      print '<p class="error">Could not retrieve the data because:<br/>' . mysql_error
        ($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
45  } // 结束查询条件语句。
46
47  mysql_close($dbc); // 关闭连接。
48
49  include('templates/footer.html'); // 包含页脚文件。
50  ?>
```

(2) 如果用户不是管理员，则终止脚本：

```
if (!is_administrator()) {
  print '<h2>Access Denied!</h2> <p class="error">You do not have permission to access
  →this page.</p>';
  include('templates/footer.html');
  exit();
}
```

这段代码几乎和add_quote.php中的代码完全一样，只是修改了浏览器上显示的标题。非管理员用户会像上一节一样被拒绝访问（参见图13-15）。

```
                <!-- BEGIN CHANGEABLE CONTENT. --><h2>All Quo
<p><b>Quote Admin:</b> <a href="edit_quote.php?id=13">Edit</a
                <a href="delete_quote.php?id=13">Delete</a></
<div><blockquote>It is the mark of an educated mind to be abl
 <strong>Favorite!</strong><p><b>Quote Admin:</b> <a href="ed
                <a href="delete_quote.php?id=9">Delete</a></p
<div><blockquote>In all affairs it's a healthy thing now and
mark on the things you have long taken for granted.</blockquo
<p><b>Quote Admin:</b> <a href="edit_quote.php?id=5">Edit</a>
                <a href="delete_quote.php?id=5">Delete</a></p
<div><blockquote>Nurture your mind with great thoughts, for y
<p><b>Quote Admin:</b> <a href="edit_quote.php?id=6">Edit</a>
                <a href="delete_quote.php?id=6">Delete</a></p
```

图13-15　该页面的HTML源代码显示了quote_id的值是如何通过URL被发送到连接页面的

(3) 包含数据库连接并定义查询：

```
include('../mysql_connect.php');
$query = 'SELECT quote_id, quote, source, favorite FROM quotes ORDER BY
→date_entered DESC';
```

这个查询返回数据库中所有记录的4列（quote_id、quote、source、favorite），返回结果按数据添加日期降序显示。

(4) 执行查询并获取结果：

```
if ($r = mysql_query($query, $dbc)) {
  while ($row = mysql_fetch_array($r)) {
```

while循环代码在第12章阐释过，不过当时并没有使用这段代码。这段代码的目的是获取所有查询语句返回的记录。

(5) 开始打印记录：

```
print "<div><blockquote>{$row['quote']}</blockquote>- {$row['source']}\n";
```

代码以<div>标签开始，将名人名言放在一对<blockquote>标签之中，随后显示名言的出处。

(6) 判断名言是否受欢迎：

```
if ($row['favorite'] == 1) {
  print ' <strong>Favorite!</strong>';
}
```

$row['favorite']的值可以是1或0。如果是1，会在名言后面显示一个明显的Favorite!标识。

(7) 添加管理链接，用于编辑和删除名人名言：

```
print "<p><b>Quote Admin:</b><a href=\"edit_quote.php?id=
→{$row['quote_id']}\">Edit</a> <->
<a href=\"delete_quote.php?id={$row['quote_id']}\">Delete</a></p></div>\n";
```

必须为每个名言创建两个链接。第一个连接到edit_quote.php，第二个连接到delete_quote.php。每个链接必须通过URL发送quote_id，类似第12章示例中的代码。

(8) 完成while循环和mysql_query()条件语句：

```
  } // 结束循环。
} else { // 没有运行查询。
  print '<p class="error">Could not retrieve the data because:<br/>' . mysql_error
  →($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
} // 结束查询条件语句。
```

(9) 断开数据库连接：

```
mysql_close($dbc);
```

(10) 完成页面：

```
include('templates/footer.html');
?>
```

(11) 将文件保存为view_quotes.php并在Web浏览器中测试。

✓提示

- 本章的一些查询语句可以使用ORDER BY子句按某个列（或值）排序，查询语句没有选择的列也可以作为排序依据。

13.9　编辑名人名言

view_quotes.php页面（和后面要创建的index.php页面）包含指向edit_quote.php的链接，用于管理员编辑名人名言。从功能上看，这个脚本与第12章创建的edit_entry.php非常类似：

(1) 脚本需要在URL中接收id值；

(2) 通过接收的id值获取记录并弹出表单（参见图13-16）；

(3) 表单提交后，会验证表单数据（只实现最基本的验证）；

(4) 如果数据通过验证，就会更新数据库中的相关记录。

本节中的代码大部分都在之前的章节阐释过。你需要学习的新东西是如何根据原有的值选中或不选中表单的复选框元素。

图13-16　表单元素会包含数据库记录中的值，并同表单一起弹出

⇛ 创建`edit_quote.php`

(1) 在文本编辑器或IDE中新建一个PHP文档，命名为`edit_quote.php`（参看脚本13-9）：

```
<?php // 脚本13-9 - edit_quote.php
define('TITLE', 'Edit a Quote');
include('templates/header.html');
print '<h2>Edit a Quotation</h2>';
```

脚本13-9　`edit_quote.php`脚本用于让管理员编辑已有记录

```
1   <?php // 脚本13-9 - edit_quote.php
2   /* 脚本编辑名人名言。*/
3
4   // 定义页面标题并包含页头文件：
5   define('TITLE', 'Edit a Quote');
6   include('templates/header.html');
7
8   print '<h2>Edit a Quotation</h2>';
9
10  // 强制只有管理员可以访问该页面：
11  if (!is_administrator()) {
12      print '<h2>Access Denied!</h2>
        <p class="error">You do not have permission to access this page.</p>';
13      include('templates/footer.html');
14      exit();
15  }
```

```
16
17  // 包含数据库连接:
18  include('../mysql_connect.php');
19
20  if (isset($_GET['id']) && is_numeric($_GET['id']) && ($_GET['id'] > 0) )
    { // Display the entry in a form:
21
22    // 定义查询。
23    $query = "SELECT quote, source, favorite FROM quotes WHERE quote_id={$_GET['id']}";
24    if ($r = mysql_query($query, $dbc)) { // 运行查询。
25
26       $row = mysql_fetch_array($r); // 返回信息。
27
28       // 创建表单:
29       print '<form action="edit_quote.php" method="post">
30          <p><label>Quote <textarea name="quote" rows="5" cols="30">' .
            htmlentities($row['quote']) . '</textarea></label></p>
31          <p><label>Source <input type="text" name="source"value="' . htmlentities
            ($row['source']) . '" /></label></p>
32          <p><label>Is this a favorite? <input type="checkbox" name="favorite"
            value="yes"';
33
34       // 检查是否选中受欢迎复选框:
35       if ($row['favorite'] == 1) {
36          print ' checked="checked"';
37       }
38
39       // 完成表单:
40       print ' /></label></p>
41         <input type="hidden" name="id" value="' . $_GET['id'] . '" />
42         <p><input type="submit" name="submit" value="Update This Quote!" /></p>
43    </form>';
44
45    } else { // 无法获取信息。
46       print '<p class="error">Could not retrieve the quotation because:<br/>' .
         mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
47    }
48
49  } elseif (isset($_POST['id']) && is_numeric($_POST['id']) && ($_POST['id'] > 0))
    { // 处理表单。
50
51    // 验证表单数据并确保安全:
52    $problem = FALSE;
53    if ( !empty($_POST['quote']) && !empty($_POST['source']) ) {
54
55      // 准备查询中使用的值:
56      $quote = mysql_real_escape_string(trim(strip_tags($_POST['quote'])), $dbc);
57      $source = mysql_real_escape_string(trim(strip_tags($_POST['source'])), $dbc);
58
59      // 创建favorite值:
60      if (isset($_POST['favorite'])) {
61         $favorite = 1;
62      } else {
63         $favorite = 0;
```

```
64        }
65
66      } else {
67        print '<p class="error">Please submit both a quotation and a source.</p>';
68        $problem = TRUE;
69      }
70
71      if (!$problem) {
72
73        // 定义查询。
74        $query = "UPDATE quotes SET quote='$quote', source='$source', favorite=
          $favorite WHERE quote_id={$_POST['id']}";
75        if ($r = mysql_query($query, $dbc)) {
76           print '<p>The quotation has been updated.</p>';
77        } else {
78           print '<p class="error">Could not update the quotation because:<br/>' .
             mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
79        }
80
81      } // 一切正常!
82
83    } else { // 没有获取id。
84       print '<p class="error">This page has been accessed in error.</p>';
85    } // 结束主条件语句。
86
87    mysql_close($dbc); // 关闭连接。
88
89    include('templates/footer.html'); // 包含页脚文件。
90    ?>
```

(2) 如果用户不是管理员则终止脚本：

```
if (!is_administrator()) {
  print '<h2>Access Denied!</h2> <p class="error">You do not have permission to access
  →this page.</p>';
  include('templates/footer.html');
  exit();
}
```

(3) 包含数据库连接：

```
include('../mysql_connect.php');
```

脚本中的两个功能（显示和处理表单），都需要数据库连接，所以这里包含数据库连接文件。

(4) 验证URL是否接收到数值型id值：

```
if (isset($_GET['id']) && is_numeric($_GET['id']) && ($_GET['id'] > 0) ) {
```

这个条件语句和第12章的类似，除此之外还会检查id值是否大于0。添加这个子句是出于脚本安全性的考虑：`is_numeric()`测试查询中的值是否安全，如果id值不可用，就不会执行查询。

(5) 定义并执行查询：

```
$query = "SELECT quote, source, favorite FROM quotes WHERE quote_id={$_GET['id']}";
if ($r = mysql_query($query, $dbc)) {
  $row = mysql_fetch_array($r);
```

查询返回记录中的3列。

(6) 创建表单：

```
print '<form action="edit_quote.php" method="post">
<p><label>Quote <textarea name="quote" rows="5" cols="30">' .
→htmlentities($row['quote']) .'</textarea></label></p>
<p><label>Source <input type="text" name="source"value="' .
→htmlentities($row['source']) .'" /></label></p>
<p><label>Is this a favorite? <input type="checkbox"
→name="favorite" value="yes"';
```

表单会返回到同一个页面。表单的开头部分是文本区，它的值是事先从数据库中检索出来的名人名言。这个值会通过`htmlentities()`函数进行转换，以确保安全。

接下来，创建了一个文本输入框，它的值是事先从数据库中检索出的名人名言出处。最后是创建是否受欢迎的复选框。注意，这个复选框目前还没有完成，接下来的第(7)步脚本就会检查复选框是否被选中。

(7) 检查是否选中受欢迎复选框：

```
if ($row['favorite'] == 1) {
  print ' checked="checked"';
}
```

如果数据库中`favorite`的值等于1，就说明管理员先前选中过受欢迎复选框。在这种情况下，需要添加一些HTML，将复选框状态置为选中。在这段代码之后，受欢迎复选框的后台HTML代码可能是：

```
<input type="checkbox" name="favorite" value="yes"
```

或（参见图13-17）

```
<input type="checkbox" name="favorite" value="yes" checked="checked"
```

```
<textarea name="quote" rows="5" cols="30">It is the mark of an educated mind to be ab
 <input type="text" name="source"value="Aristotle" /></label></p>
s a favorite? <input type="checkbox" name="favorite" value="yes" checked="checked" />
den" name="id" value="9" />
```

图13-17　该页面的HTML源代码，在首次访问时，显示先前选中的受欢迎复选框

(8) 完成表单：

```
print ' /></label></p>
<input type="hidden" name="id" value="' . $_GET['id'] . '" />
<p><input type="submit" name="submit" value="Update This Quote!" /></p>
</form>';
```

`print`语句首先关闭复选框。接下来，表单必须将`id`值存储在隐藏的文本框中，以便在提交表单时使用。

(9) 如果没有检索到记录，提示错误信息：

```
} else { // 无法获取信息。
print '<p class="error">Could not retrieve the quotation
→ because:<br/>' . mysql_ error($dbc) . '.</p><p>The
```

```
→ query being run was: ' . $query . '</p>';
}
```

(10) 检查表单提交情况：

```
} elseif (isset($_POST['id']) && is_numeric($_POST['id']) && ($_POST['id'] > 0)) {
```

这个条件语句是脚本第二个部分（处理表单提交）的起始处。验证的方式与第(4)步类似，但这里引用的是`$_POST['id']`，而不是`$_GET['id']`。

(11) 验证表单数据：

```
$problem = FALSE;
if ( !empty($_POST['quote']) &&!empty($_POST['source']) ) {
```

表单验证的方式同add_quote.php脚本的验证方式相同。

(12) 准备值以便在查询中使用：

```
$quote = mysql_real_escape_string (trim(strip_tags($_POST['quote'])), $dbc);
$source = mysql_real_escape_ string(trim(strip_tags($_POST['source'])), $dbc);
if (isset($_POST['favorite'])) {
  $favorite = 1;
} else {
  $favorite = 0;
}
```

这段代码同add_quote.php脚本里使用的相同。

(13) 如果表单没有完成，显示错误信息：

```
} else {
  print '<p class="error">Please submit both a quotation and a source.</p>';
  $problem = TRUE;
}
```

(14) 如果没有问题发生，更新数据库：

```
if (!$problem) {
  $query = "UPDATE quotes SET quote='$quote', source='$source', favorite=$favorite
  →WHERE quote_id={$_POST['id']}";
  if ($r = mysql_query($query, $dbc)) {
  print '<p>The quotation has been updated.</p>';
```

查询更新每条记录的3列值。两个字符串值用单引号引起来（在查询内），而`$favorite`是数值型，不需要使用单引号。

`WHERE`子句指明要更新的记录，是最重要的部分。

最后，打印更新成功的信息（参见图13-18）。



图13-18　成功编辑记录后的页面

(15) 如果更新失败，显示错误信息：

```
} else {
  print '<p class="error">Could not update the quotation because:<br/>' . mysql_
  → error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
}
```

(16) 完成条件语句：

```
  } // 一切正常!
} else { // No ID set.
  print '<p class="error">This page has been accessed in error.</p>';
} // 结束主条件语句。
```

else子句处理没有通过GET或POST方式收到有效id值的情况。

(17) 关闭数据库连接并完成页面：

```
mysql_close($dbc);
include('templates/footer.html');
?>
```

(18) 保存文件并在Web浏览器中测试（单击view_quotes.php上的链接）。

13.10　删除名人名言

删除已有名人名言的脚本类似第12章的delete_entry.php脚本。第一次访问时，如果通过URL传递的是一个有效的id，就会显示要删除的名人名言记录（参见图13-19）。如果管理员单击了提交按钮，表单会返回相同的页面，并从数据库中删除这条记录（参见图13-20）。

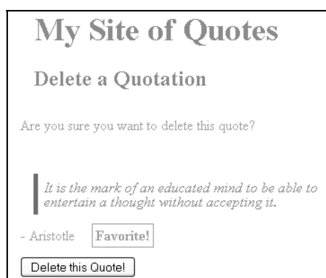


图13-19　删除记录的第一步是确认要删除的记录



图13-20　表单提交后，名人名言会被删除，同时打印成功信息

⇒ 创建delete_quote.php

(1) 在文本编辑器或IDE中新建一个PHP文档，命名为delete_quote.php（参看脚本13-10）：

```
<?php // 脚本13-10 - delete_quote.php
define('TITLE', 'Delete a Quote');
include('templates/header.html');
print '<h2>Delete a Quotation</h2>';
```

脚本13-10　delete_quote.php脚本用于让管理员删除已有记录

```
1   <?php // 脚本13-10 - delete_quote.php
2   /* 脚本删除名人名言。*/
3
4   // 定义页面标题并包含页头文件：
5   define('TITLE', 'Delete a Quote');
6   include('templates/header.html');
7
8   print '<h2>Delete a Quotation</h2>';
9
10  // 强制只有管理员可以访问该页面：
11  if (!is_administrator()) {
12     print '<h2>Access Denied!</h2>
       <p class="error">You do not have permission to access this page.</p>';
13     include('templates/footer.html');
14     exit();
15  }
16
17  // 包含数据库连接：
18  include('../mysql_connect.php');
19
20  if (isset($_GET['id']) && is_numeric($_GET['id']) && ($_GET['id'] > 0) ) {
    // 在表单中显示名人名言：
21
22    // 定义查询：
23    $query = "SELECT quote, source, favorite FROM quotes WHERE quote_id=
      {$_GET['id']}";
24    if ($r = mysql_query($query, $dbc)) { // 运行查询。
25
26      $row = mysql_fetch_array($r); // 返回信息。
27
28      // 创建表单：
29      print '<form action="delete_quote.php" method="post">
30      <p>Are you sure you want to delete this quote?</p>
31      <div><blockquote>' . $row['quote'] . '</blockquote>- ' . $row['source'];
32
33      // 检查是否选中受欢迎复选框：
34      if ($row['favorite'] == 1) {
35          print ' <strong>Favorite!</strong>';
36      }
37
38      print '</div><br/><input type="hidden" name="id" value="' . $_GET['id'] . '" />
39      <p><input type="submit" name="submit" value="Delete this Quote!" /></p>
40      </form>';
41
42    } else { // 无法获取消息。
43      print '<p class="error">Could not retrieve the quote because:<br/>' .
        mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
44    }
45
46  } elseif (isset($_POST['id']) && is_numeric($_POST['id']) && ($_POST['id'] > 0) )
    { // 处理表单。
47
```

```
48     // 定义查询：
49     $query = "DELETE FROM quotes WHERE quote_id={$_POST['id']} LIMIT 1";
50     $r = mysql_query($query, $dbc); // 执行查询。
51
52     // 打印结果：
53     if (mysql_affected_rows($dbc) == 1) {
54       print '<p>The quote entry has been deleted.</p>';
55     } else {
56       print '<p class="error">Could not delete the blog entry because:<br/>' .
         mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
57     }
58
59   } else { // 没有获取id。
60     print '<p class="error">This page has been accessed in error.</p>';
61   } // 结束主条件语句。
62
63   mysql_close($dbc); // 关闭连接。
64
65   include('templates/footer.html');
66   ?>
```

(2) 如果用户不是管理员则终止脚本：

```
if (!is_administrator()) {
  print '<h2>Access Denied!</h2> <p class="error">You do not
  ➝have permission to access this page.</p>';
  include('templates/footer.html');
  exit();
}
```

(3) 包含数据库连接：

```
include('../mysql_connect.php');
```

(4) 验证ID值是从URL中接收的：

```
if (isset($_GET['id']) && is_numeric($_GET['id']) && ($_GET['id'] > 0) ) {
```

这个条件语句与edit_quote.php中使用的一样。

(5) 检索要删除的数据：

```
$query = "SELECT quote, source, favorite FROM quotes WHERE quote_id={$_GET['id']}";
if ($r = mysql_query($query, $dbc)) { $row = mysql_fetch_array($r);
```

从数据库中返回检索记录的3个字段。因为这个脚本只会处理一条记录，所以只在循环外调用一次mysql_fetch_array()函数。

(6) 开始创建表单：

```
print '<form action="delete_quote.php" method="post">
<p>Are you sure you want to delete this quote?</p>
<div><blockquote>' . $row['quote'] .'</blockquote>- ' .
$row['source'];
```

表单主要用于显示名人名言。

(7) 如果名人名言是受欢迎的，显示Favorite!标记：

```
if ($row['favorite'] == 1) {
  print ' <strong>Favorite! </strong>';
}
```

这段代码与view_quotes.php页面中的一样，显示这是受欢迎的名言。

(8) 完成表单：

```
print '</div><br/>
<input type="hidden" name="id" value="' . $_GET['id'] . '" />
<p><input type="submit" name="submit" value="Delete this Quote!" /></p>
</form>';
```

表单必须包含隐含的输入框，在提交时，这个输入框会将名言的id传回页面。

(9) 完成mysql_query()条件语句：

```
} else { // 无法获取信息。
  print '<p class="error">Could not retrieve the quote
  →because:<br/>' . mysql_ error($dbc) . '.</p><p>The
  →query being run was: ' . $query . '</p>';
}
```

如果查询失败、MySQL出错或查询本身出错，则会显示调试建议。

(10) 检查表单提交情况：

```
} elseif (isset($_POST['id']) && is_numeric($_POST['id']) && ($_POST['id'] > 0) )) {
```

这个条件语句开启了脚本的第二个部分（处理表单提交）。验证方式和第(4)步相同，但是这里引用的是$_POST['id']，而不是$_GET['id']。

(11) 删除记录：

```
$query = "DELETE FROM quotes WHERE quote_id={$_POST['id']} LIMIT 1";
$r = mysql_query($query, $dbc);
```

DELETE语句会删除记录。WHERE子句指明要删除的特定记录，LIMIT 1子句是另一个限制条件。

(12) 报告结果：

```
if (mysql_affected_rows($dbc) == 1) {
  print '<p>The quote entry has been deleted.</p>';
} else {
  print '<p class="error">Could not delete the blog entry because:<br/>' . mysql_
  →error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
}
```

如果语句执行成功，就会删除一条记录，并向用户显示信息（参见图13-20）。

(13) 完成条件语句：

```
} else { // 没有获取id。
  print '<p class="error">This page has been accessed in error.</p>';
} // 结束主条件语句。
```

else子句处理没有通过GET或POST方式收到有效id值的情况。

(14) 关闭数据库连接并完成页面：

```
mysql_close($dbc);
include('templates/footer.html');
?>
```

(15) 保存文件并在Web浏览器中测试（单击`view_quotes.php`上的链接）。

13.11　创建主页

最后，我们需要创建主页。对于这个网站来说，主页面是唯一一个面向公众的页面。主页会显示一条名人名言，这个名言可以是以下几种类型：

- 最新添加的名人名言（默认）；
- 随机的名人名言；
- 随机的受欢迎的名人名言。

要实现这个效果，链接将在URL中传送不同的值到这个脚本（参见图13-21）。

如果用户是管理员，这个脚本还需要在当前显示的名人名言下面显示管理链接（编辑链接和删除链接）（参见图13-22）。



图13-21　通过URL传送不同的值以显示不同的名言



图13-22　当管理员查看主页时，会显示更多管理链接

⇒ 创建index.php

(1) 在文本编辑器或IDE中新建一个PHP文档，命名为`index.php`（参看脚本13-11）：

```
<?php // 脚本13-11 - index.php
```

脚本13-11　网站的主页一次显示一条名人名言，在管理员登录时会显示管理链接

```
1   <?php // 脚本13-11 - index.php
2   /* 这是网站的主页。包含以下内容：
3   - 最新名人名言(默认)
4   - 或，随机名人名言
5   - 或，随机受欢迎的名人名言 */
6
7   // 包含页头文件：
8   include('templates/header.html');
```

```
9
10 // 包含数据库连接:
11 include('../mysql_connect.php');
12
13 // 定义查询……
14 // 根据URL传递过来的值更改查询方式:
15 if (isset($_GET['random'])) {
16     $query = 'SELECT quote_id, quote, source, favorite FROM quotes ORDER BY RAND()
       DESC LIMIT 1';
17 } elseif (isset($_GET['favorite'])) {
18    $query = 'SELECT quote_id, quote, source, favorite FROM quotes WHERE favorite=1
       ORDER BY RAND() DESC LIMIT 1';
19 } else {
20     $query = 'SELECT quote_id, quote, source, favorite FROM quotes ORDER BY
       date_entered DESC LIMIT 1';
21 }
22
23 // 运行查询:
24 if ($r = mysql_query($query, $dbc)) {
25
26     // 返回查询结果:
27     $row = mysql_fetch_array($r);
28
29     // 打印查询结果:
30     print "<div><blockquote>{$row['quote']}</blockquote>- {$row['source']}";
31
32     // 判断是否为受欢迎的?
33     if ($row['favorite'] == 1) {
34        print ' <strong>Favorite!</strong>';
35     }
36
37     // 完成div标签:
38     print '</div>';
39
40     // 如果管理员登录，显示管理员链接:
41     if (is_administrator()) {
42        print "<p><b>Quote Admin:</b> <a href=\"edit_quote.php?id={$row['quote_id']}
          \">Edit</a> <->
43        <a href=\"delete_quote.php?id={$row['quote_id']}\">Delete</a>
44        </p>\n";
45     }
46
47 } else { // 没有运行查询。
48      print '<p class="error">Could not retrieve the data because:<br/>' .
        mysql_error($dbc) . '.</p><p>The query being run was: ' . $query . '</p>';
49 } // End of query IF.
50
51 mysql_close($dbc); // 关闭连接。
52
53 print '<p><a href="index.php">Latest</a> <-> <a href="index.php?random=true">
   Random</a> <-> <a href="index.php?favorite=true">Favorite</a><p>';
54
55 include('templates/footer.html'); // 包含页脚文件。
56 ?>
```

(2) 包含头文件：

```
include('templates/header.html');
```

主页不需要自定义标题，因此在包含头文件之前不需要定义常量。

(3) 包含数据库连接：

```
include('../mysql_connect.php');
```

(4) 开始定义要运行的查询：

```
if (isset($_GET['random'])) {
  $query = 'SELECT quote_id, quote, source, favorite FROM
  →quotes ORDER BY RAND() DESC LIMIT 1';
```

如果$_GET['random']变量已设置，用户单击某个链接请求一条随机名人名言。无论这个变量的值是什么，只要设置了即可。

所有的查询都会返回来自同一行的4个列值（quote_id、quote、source和favorite）。如果只检索一行，使用LIMIT 1子句即可。

使用ORDER BY RAND()子句，在MySQL中选择一个随机的行。这个代码使用MySQL的RAND()函数，RAND是random的缩写。这个查询首先以随机顺序选择所有记录，之后返回这个集合中的首条记录。

(5) 定义查询，该查询选择一条随机的受欢迎的记录：

```
} elseif (isset($_GET['favorite'])) {
  $query = 'SELECT quote_id, quote, source, favorite FROM
  →quotes WHERE favorite=1 ORDER BY RAND() DESC LIMIT 1';
```

这个查询与第(4)步的一样，但这里使用WHERE子句来选择那些favorite值等于1的记录。

(6) 定义默认查询：

```
} else {
  $query = 'SELECT quote_id, quote, source, favorite FROM
  →quotes ORDER BY date_entered DESC LIMIT 1';
}
```

如果没有通过URL传送任何值，主页会显示最新添加的名人名言。实现方法是按名言添加时间的降序排序所有记录，并限制返回一条记录。

(7) 执行查询并获取返回记录：

```
if ($r = mysql_query($query, $dbc)) {
    $row = mysql_fetch_array($r);
```

(8) 打印名人名言：

```
print "<div><blockquote>{$row['quote']}</blockquote>- {$row['source']} ";
```

这个代码同view_quotes.php脚本中的一样，但只需要使用一次。

(9) 如果名言是受欢迎的，就会在显示标识之后结束<div>标签:

```
if ($row['favorite'] == 1) {
  print ' <strong>Favorite!</strong>';
```

```
}
print '</div>';
```

这个条件语句与delete_quote.php和view_quotes.php脚本中的一样。

(10) 如果用户是管理员，创建用于编辑和删除这条记录的链接：

```
if (is_administrator()) {
  print "<p><b>Quote Admin:</b> <a href=\"edit_quote.php?
  →id={$row['quote_id']}\">Edit </a> - + | + -
  <a href=\"delete_quote.php? id={$row['quote_id']}\"> Delete</a>
  </p>\n";
}
```

如果用户是管理员，这个页面就会添加连接到编辑和删除脚本的链接。这些链接的值与view_quotes.php脚本中一样。

(11) 如果查询不能运行，打印错误信息：

```
} else { // 没有运行查询。
  print '<p class="error">Could not retrieve the data
  →because:<br/>' . mysql_error ($dbc) . '.</p><p>The query
  →being run was: ' . $query .'</p>';
} // 结束查询条件语句。
```

以上代码是出于调试目的而设计的。你不应该向公众暴露MySQL错误信息或引起错误的查询信息。

(12) 关闭数据库连接：

```
mysql_close($dbc);
```

之后不需要再使用数据库连接了，因此在这里关闭它。

(13) 创建连接到其他页面的链接：

```
print '<p><a href="index.php"> Latest</a> <-> <a href=
  →"index.php?random=true"> Random</a> <-> <a href=
  →"index.php?favorite=true"> Favorite</a><p>';
```

添加3个面向公众的链接，每个链接都指向这个脚本。第1个链接不会通过URL传送任何值，它会一直显示最近添加的名言。第2个链接，通过URL传递一个随机值，这会触发第(4)步的查询。第3个链接，通过URL传递受欢迎值，这会触发第(5)步的查询，获取一个随机的受欢迎记录。

(14) 包含页脚文件并完成页面：

```
include('templates/footer.html');
?>
```

(15) 保存文件并在Web浏览器中测试（参见图13-23）。

✓提示

- 通常主页是首先要编写的脚本，而不是在最后。但在这个例子中，我想展示这个页面是如何一步步创建起来的。



图13-23　最近添加的名人名言（注意URL）

13.12　回顾和实践

如果你对本节中的内容存有疑问，可以在本书的论坛上提问，网站地址：www.LarryUllman.com/forum/。

13.12.1　回顾

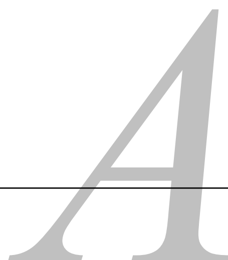
- ❑ 如何调用`is_administrator()`函数检查有不同值的同一个cookie（名字为Samuel）？如何检查有不同值的不同cookie（名字不为Samuel）？
- ❑ 为什么在头文件中引用样式表时使用的是`css/style.css`而不是`../css/style.css`？还可以用什么方式引用样式表？
- ❑ 为什么`login.php`脚本以那种方式构建？如何以更加线性的方式来组织这个脚本。
- ❑ 这个网站还可以使用哪些用户自定义函数？提示：关键在于找到重复使用的代码。

13.12.2　实践

- ❑ 将登录表单修改成粘性表单。
- ❑ 定义登录验证信息（cookie名和值），将它作为配置文件中的常量。接下来在每个页面中包含配置文件并使用这些常量创建、删除和确认cookie的值。
- ❑ 限制cookie的过期时间为15分钟，接下来在适当的情况下（即如果cookie存在），在每个页面上再次发送cookie。
- ❑ 使用session代替cookie。
- ❑ 将`add_quote.php`和`edit_quote.php`表单修改成粘性表单。
- ❑ 修改`view_quotes.php`脚本，让管理员可以以不同的顺序显示名言。提示：创建类似`index.php`脚本中返回页面的链接，然后相应修改查询语句。
- ❑ 在将这个网站放到实际服务器上之前（你应该做这件事），更新所有代码，以确保非管理员不会看到任何MySQL错误。

附录 A

安装和配置



运行本书示例，需要3种技术（工具）：MySQL（数据库应用程序）、PHP（脚本语言）和Web服务器应用程序（使PHP得以运行）。附录A将介绍如何在Windows和Macintosh这两个不同的平台上安装这些工具。如果读者使用的是Web站点主机，它可能已经提供了这些工具，但由于这些产品都是免费的并且易于安装，因此也可以将它们安装在自己的计算机上。

在介绍了安装之后，本附录还将介绍使用MySQL和配置PHP的基础知识。PHP和MySQL的手册都相当详细地介绍了它们的安装和配置方法。如果遇到问题，可以详细地阅读这些手册以寻求解决方法。

A.1　在 Windows 上安装

虽然你可以在Windows系统上分别安装Web服务器（例如，Apache、Abyss或IIS）、PHP和MySQL，但是我强烈建议你使用多合一安装包。这样做更容易，也不太会出现问题。

用于Windows的集成安装程序有很多。XAMPP（www.apachefriends.org）和WAMP（www.wampserver.com）是最经常提到的两种集成安装程序。在附录A中，本书将使用XAMPP，它可以运行在Windows 2000、Windows 2003、Windows XP和Vista上。（XAMPP网站上并没有提及Windows 7，但它可以运行在Windows 7上。）

除了Apache、PHP和MySQL之外，XAMPP还安装了：

- PEAR——一个PHP代码库；
- phpMyAdmin——有Web界面的MySQL服务器；
- 一个邮件服务器（用于发送邮件）；
- 大量有用的扩展。

本书在编写时，XAMPP（版本1.7.3）会同时安装PHP 5.31、MySQL 5.1.41、Apache 2.2.14和phpMyAdmin 3.2.4。

下面本书将逐步介绍安装过程。请注意，如果遇到任何问题，都可以访问本书的支持论坛（www.LarryUllman.com/forum/），但向XAMPP站点求助可能会更好一些（毕竟这是他们的产品）。另外，安装程序可以工作得很好而且使用也很简单，所以本书没有详细地介绍安装过程中的每一个步骤，而是着重于更重要的需要考虑的事情。

防　火　墙

防火墙保护通过端口的通信（端口是计算机上的访问点）。Windows自XP Service Pack 2开始提供内置的防火墙。也可以下载并安装第三方防火墙。防火墙改善了计算机的安全性，却影响了由XAMPP提供的Apache、MySQL和其他工具的运行能力，因为它们都要使用端口进行通信。

在首次运行XAMPP时，如果看到安全提示，说明防火墙正在阻止Apache、MySQL或其他应用程序启动，请选择Unblock（不阻止）。你也可以手动配置防火墙（例如，在Windows XP上通过“控制面板”->“安全中心”来完成）。需要打开的端口有：80（Apache）、3306（MySQL）和25（Mercury邮件服务器）。如果在启动或访问这些端口时有问题，请禁用防火墙并检查它是否凑效。如果问题解除，则表明是防火墙引发的问题，需要重新配置。

需要澄清的是，防火墙不只存在于Windows上，但是根据本附录的说明，Windows用户比其他平台的用户更容易出现由防火墙引发的问题。

⇛ 在Windows上安装XAMPP

(1) 从www.apachefriends.org下载Windows版XAMPP的最新版本。

需要进行几次单击才能找到下载区，最终你会到达一个类似于图A-1那样的区域。然后单击EXE，这正是我们需要的东西。

(2) 在计算机上，双击下载的文件以启动安装过程。

(3) 出现提示信息后（图A-2），将XAMPP安装到Program Files以外的目录中。

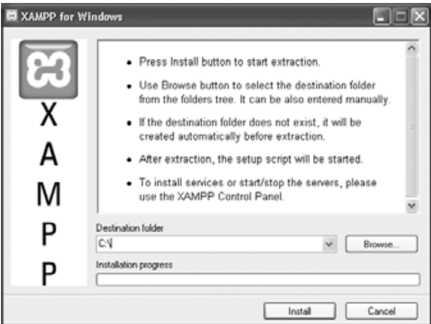
由于在Windows Vista上会遇到权限问题，因此不能将其安装到Program Files目录中。本书建议将其安装到根目录（如`C:\`）中。

无论决定将程序安装到哪里，都要注意其安装位置，在本附录中会多次提到这一点。

XAMPP for Windows 1.7.3, 2009/12/23

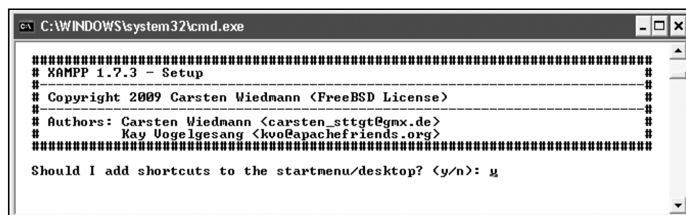
Version	Size	Content
XAMPP Windows 1.7.3 [Basic package]		Apache 2.2.14 (IPv6 enabled), MySQL 5.1.41 + PBXT engine, PHP 5.3.1, OpenSSL 0.9.8l, phpMyAdmin 3.2.4, XAMPP Control Panel 2.5.8, XAMPP CLI Bundle 1.6, Webalizer 2.21-02, Mercury Mail Transport System v4.72, msmtp 1.4.19, FileZilla FTP Server 0.9.33, SQLite 2.8.17, SQLite 3.6.20, ADOdb 5.10, eAccelerator 0.9.6-rc1, Xdebug 2.0.6-dev, Ming 0.4.3 For Windows 2000, XP, Vista, 7. See README
EXE	51 MB	Self-extracting RAR archive MD5 checksum: 3635a1c0baf15e8a019009e6c1225389
ZIP	100 MB	ZIP archive MD5 checksum: 0fe7f440a7d3af7c06981570f764d246
XAMPP Windows 1.7.3 [Upgrade 1.7.2 to 1.7.3]		
EXE	45 MB	Self-extracting RAR archive MD5 checksum: 414cb9b594f90ac9257a193c6fc6057a
ZIP	89 MB	ZIP archive MD5 checksum: 985d0e704bf543079e626f4adb54e9ad

图A-1　从Apache Friends网站获取最新的Windows版本的安装程序



图A-2　选择XAMPP的安装位置

(4) 你还可以在桌面或开始菜单中创建快捷方式（参见图A-3）。



图A-3　本书建议使用的XAMPP选项

在安装过程中，XAMPP会打开了一个控制台窗口，这个窗口会提示一些其他选项。

(5) 出现提示时，可以选择XAMPP的安装路径。

(6) 出现提示时，需要提供XAMPP的安装盘符。

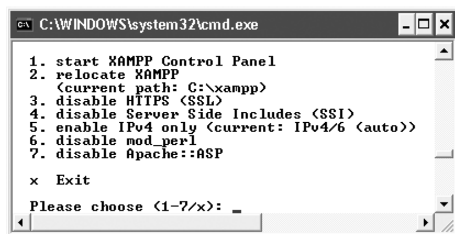
(7) 继续完成后面的步骤，阅读之后按Enter（或Return）前进。

随后，安装程序会提示安装成功，接下来安装程序会要求你设置PHP配置中的时区等内容，按照提示做就行了。

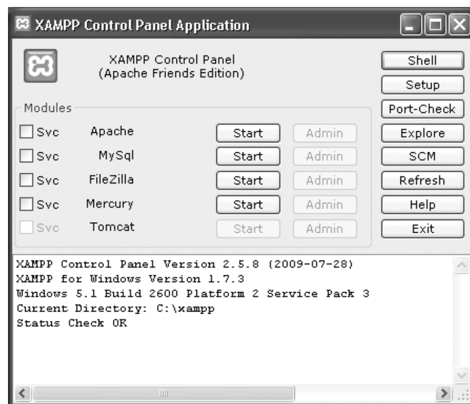
(8) 安装完成后（参见图A-4），按1键启动控制面板。

安装完成后，在控制台窗口中还会出现几个选项。一个选项是启动控制面板，你也可以单击桌面或开始菜单中的快捷方式启动它，但前提是你已经在第(4)步创建了它们。

(9) XAMPP控制面板可以启动、停止和配置XAMPP（参见图A-5）。



图A-4　安装完成



图A-5　如果运行了某种防火墙，在启动Apache或其他应用程序时可能会看到类似的消息。参见框注“防火墙”

(10) 使用控制面板启动Apache、MySQL和Mercury。

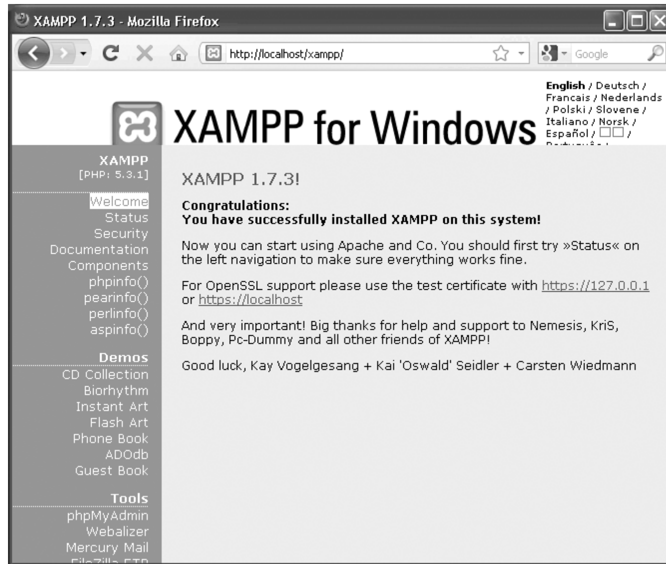
本书中所有的示例都要启动Apache。完成第12章的内容需要启动MySQL。Mercury是XAMPP安装的邮件服务器。如果要使用PHP发送Email就要启动它（详见第8章）。

(11) 立即为MySQL用户设置密码。

本章稍后将介绍如何完成这一步。

✓提示

- ❑ XAMPP的控制面板有很多管理链接，它们可以连到不同的页面上（你自己的服务器上）或其他资源上（参见图A-6）。



图A-6　控制面板中链接的XAMPP基于Web的前导页面

- ❑ 请参见本章后面的A.3节，学习如何通过编辑`php.ini`文件来配置PHP。
- ❑ Web根目录（放置PHP脚本以供测试的地方）是XAMPP安装目录中的`htdocs`文件夹。这里安装的目录是C:\xampp\htdocs。

A.2　在 Mac OS X 上安装

由于那些现成的安装包，在Mac OS X上安装MySQL和PHP曾经（也许仍然）非常简单。Mac OS X已经使用Apache作为其Web服务器了，而且还安装了PHP的某个版本，只是没有启用。感谢Marc Liyanage（www.entropy.ch），他为了支持Mac操作系统做了很多工作，因而更多PHP的当前版本和功能丰富的版本都能够很容易地安装。

本书称安装“曾经”并且“也许仍然”简单，是因为这取决于所使用的硬件和操作系统。在编写本书时，最新的Mac使用Intel 64位处理器。这些计算机可以运行32位和64位软件。为Mac OS X 10.5（Leopard）构建的Apache是64位版本的（假设计算机支持），这意味着PHP以及大量由PHP使用的库也必须是64位版本的，这并不容易做到。

由于在Mac OS X 10.5上使用64位版本的Apache（如果可以的话）会出现兼容性问题，所以本书决定介绍一种更为常见的、非常简单的方法，并建议使用MAMP（www.mamp.info）集成安装程序。它既是免费的又具有官方版本，易于使用，而且不会影响到在操作系统中内置的

Apache。

除了Apache、PHP和MySQL，MAMP还会安装phpMyAdmin（基于Web界面的MySQL）以及大量有用的PHP扩展。在编写本书时，MAMP（版本1.9.4）安装了PHP 5.2.13和5.3.2、MySQL 5.1.44 Apache 2.0.63以及phpMyAdmin 3.2.5。

下面本书将逐步介绍安装过程。请注意，如果遇到任何问题，都可以访问本书的支持论坛，但向MAMP站点求助可能会更好一些（毕竟这是他们的产品）。另外，安装程序可以工作得很好并且易于使用，所以本书没有详细地介绍安装过程中的每一个步骤，而是着重于更重要的需要考虑的事情。

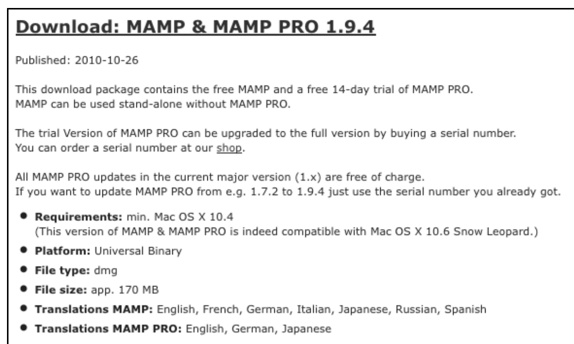
⇛ 在Mac OS X上安装MAMP

(1) 从www.mamp.info下载MAMP的最新版本。

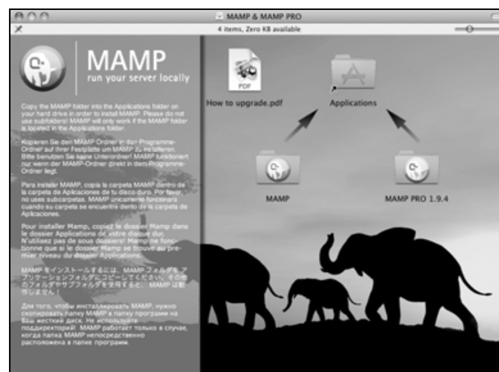
在首页单击Download，然后单击MAMP & MAMP PRO 1.9.4（参见图A-7）。（随着MAMP新版本的发布，链接和文件名也会随之变化。）

两种产品的下载文件是一样的。实际上，MAMP Pro只是有一个更好的控制和自定义MAMP软件的界面。

(2) 在计算机上，双击下载的文件，安装磁盘镜像（参见图A-8）。



图A-7　从www.mamp.info中的这个页面里下载MAMP



图A-8　下载的MAMP磁盘镜像中的内容

(3) 将MAMP文件夹从磁盘镜像复制到`Applications`文件夹。

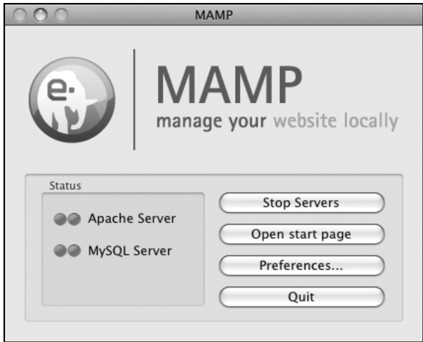
如果更喜欢商业版的MAMP PRO，请复制相应的文件夹（由于它是连接到MAMP的界面，因此需要两个文件夹），它有一个14天免费试用的版本。

不管选择的是哪个文件夹，请注意必须将其放置在`Applications`文件夹中，而不能放在一个子文件夹或计算机上的其他目录中。

(4) 打开Applications/MAMP（或Applications/MAMP PRO）文件夹。

(5) 双击MAMP（或MAMP PRO）应用程序，启动程序（参见图A-9）。

需要花一小会儿时间来启动服务器，然后就会看到如图A-9所示的结果（MAMP）或参见图A-10所示的结果（MAMP PRO）。



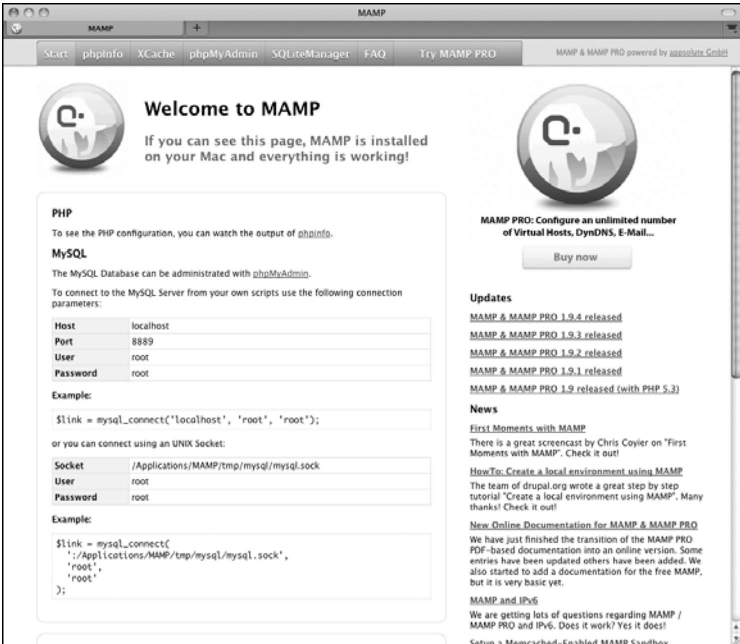
图A-9　MAMP应用程序，用于控制和配置Apache、PHP和MySQL



图A-10　MAMP PRO应用程序，用于控制和配置Apache、PHP和MySQL等

当启动MAMP时，默认Web浏览器会打开起始页（参见图A-11）。通过这个页面可以浏览到正在运行的PHP版本以及它的配置，还有使用了phpMyAdmin的MySQL数据库界面。

如果是MAMP PRO，你可以单击WebStart按钮访问这个界面（参见图A-10）。



图A-11　MAMP起始页

(6) 要启动、停止和配置MAMP，请使用MAMP或MAMP PRO应用程序（参见图A-9和图A-10）。

关于应用程序本身没有什么需要多说的（这是一个很好的程序），但如果单击Perferences，则可以调整应用程序的行为（参见图A-12），从而设置要运行的PHP版本，等等。

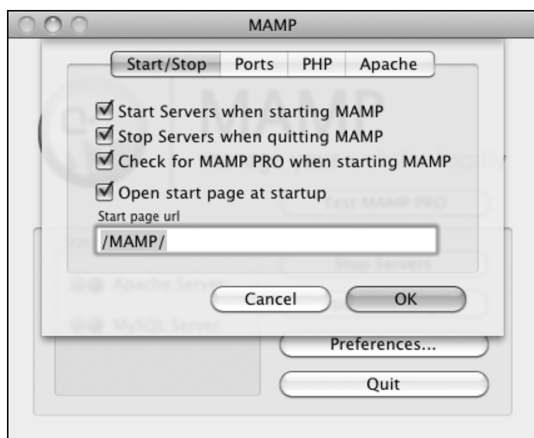
MAMP PRO也可以创建不同的虚拟主机（即不同的网站），可以调整Apache配置和运行方式，可以使用动态的DNS以及修改Email的发送方式，等等。

(7) 立即为MySQL根用户设置密码。

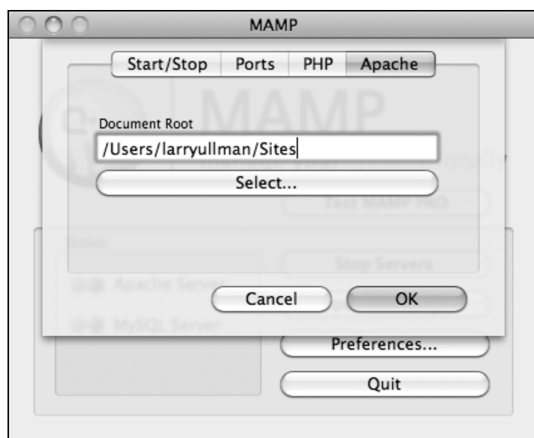
本章稍后将介绍如何进行设置。

✓提示

- ❑ 就我个人而言，我非常喜欢使用MAMP，因为它是免费的。我也不喜欢花钱的东西，但我发现花点钱买一个MAMP PRO还是很划算的。
- ❑ 参见本章最后的A.3节，学习如何通过编辑`php.ini`文件来配置PHP。
- ❑ 你可能希望将Apache Document Root（参见图A-13）设置为主文件夹中的`Sites`目录。这样做之后，请确保备份Web文档和其他文件。（此时正在执行常规备份，对吗？）
- ❑ MAMP还提供了一个Dashboard小工具，用于管理Apache和MySQL服务器。
- ❑ Web根目录（用于放置PHP脚本以进行测试的地方）是MAMP安装目录下的`htdocs`文件夹。这里安装的目录是Applications/MAMP/htdocs。



图A-12　这5个选项决定着当启动和停止MAMP应用程序时会发生什么



图A-13　MAMP允许修改Web文档放置的位置

A.3　PHP 配置

自己安装PHP的一个好处在于，你可以按照自己的喜好来进行配置。PHP如何运行可由`php.ini`文件决定，该文件通常在PHP安装时创建。

需要考虑调整的两个重要配置是display_errors和error_reporting（均在第3章中讨论过）。要修改任何设置，请打开PHP配置文件，按照需要对其进行编辑，然后保存该文件，并重启Web服务器。

更改PHP配置

(1) 在Web浏览器中执行调用phpinfo()函数的脚本（参见图A-14）。

phpinfo()函数在第1章介绍过，它可以显示PHP安装信息。

http://localhost/xampp/phpinfo.php

PHP Version 5.3.1

System	Windows NT WINMBP 5.1 build 2600 (Windows XP Professional Service Pack 3) i586
Build Date	Nov 20 2009 17:20:57
Compiler	MSVC6 (Visual C++ 6.0)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	*no value*
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,TS,VC6

图A-14　调用phpinfo()函数的部分输出

(2) 在浏览器输出中，查找加载的配置文件。

这个文件后面的值就是配置文件的地址。地址可能像这样：C:\xampp\php\php.ini或/Library/ApplicationSupport/appsolute/MAMP PRO/conf/php.ini。

如果配置文件后面没有值，就说明你的服务器上没有php.ini文件。这种情况下，你需要从www.php.net上下载PHP源代码，从中找到这个文件。

(3) 在任意文本编辑器中打开php.ini文件。

(4) 按照需要修改设置。

根据你使用的操作系统，你可能需要管理员身份或输入密码才可以修改该文件。

该文件包含很多指令。通过在前面加分号，可以将某些行注释掉（使其无效）。

(5) 保存php.ini文件。

(6) 重启Web服务器。

无需重启整个计算机，只需重启Web服务器（如Apache）即可。

✓提示

- 还可以使用`phpinfo()`脚本来检查变更的配置是否起作用了。
- 如果编辑了`php.ini`文件并重启了Web服务器，而这些修改没有起作用，请确认编辑了适当的`php.ini`文件（所用的计算机上可能有多个该文件）。
- 在Mac OS X上的MAMP PRO会为`php.ini`维护一个模板，必须在MAMP PRO中编辑`php.ini`文件。在使用MAMP PRO时，如果修改PHP配置，可以单击File >Edit Template > PHP *X.X.X* php.ini。

启用邮件

只有运行PHP的计算机能够访问sendmail或其他邮件服务器时，才能使用`mail()`函数。一种启用`mail()`函数的方式是设置`php.ini`文件中的`smtp`值（仅用于Windows）。如果能使用Internet提供商提供的SMTP地址，则可以正常使用。遗憾的是，如果ISP的SMTP服务器要求验证，就不能使用这种方式。

对于Windows，有很多免费的SMTP服务器，如Mercury。它会随XAMPP一起安装，或者如果不使用XAMPP，也可以单独安装它。

Mac OS X本身安装有邮件服务器（postfix或sendmail），但需要启用它们。搜索一下Google便可以找到在Mac OS X上手动启用邮件服务器的操作说明。

或者，可以搜索PHP代码库，学习如何使用需要验证的SMTP服务器。

A.4　MySQL 界面

在第12章和第13章，我们使用PHP脚本与MySQL进行交互。如第12章所述，最好的调试方式是使用其他应用程序来执行SQL命令。了解如何使用独立的MySQL界面非常重要。这里会介绍两种最常见的界面。

A.4.1　使用MySQL客户端

MySQL软件包含了一个重要的工具，称作MySQL*客户端*。该应用程序提供了与MySQL服务器进行通信的界面。这是一个命令行工具，它在Linux和Mac OS X上必须使用Terminal应用程序访问，而在Windows上则必须使用命令提示符（DOS）。

⇛ 使用MySQL客户端

(1) 确保MySQL服务器正在运行。

(2) 找到MySQL的`bin`目录。

要连接到客户端，需要知道它的位置。MySQL客户端可以在安装目录的`bin`目录中找到。本

书将使用最常用的安装目录。

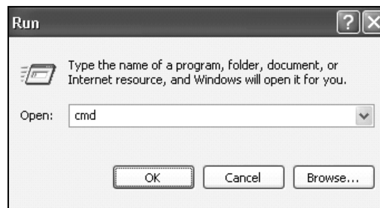
如果是自己安装的MySQL，则客户端的位置取决于软件的安装位置，但很可能是`C:\mysql\bin\mysql`（Windows）或`/usr/local/mysql/bin/mysql`（Mac OS X and Unix）。

如果在Windows上使用了XAMPP，则可能是`C:\xampp\mysql\bin\mysql`（假设将XAMPP安装到了`C:\`）。如果在Mac OS X上安装了MAMP，则MySQL的目录是`/Applications/MAMP/Library/bin/mysql`。

(3) 访问命令提示符。

在Mac OS X和Unix上，可以运行Terminal应用程序。在Mac OS X上，可以在`Applications/Utilities`文件夹中找到它。

在Windows上，单击Start菜单，选择Run，然后输入cmd并按回车键（图A-15）。



图A-15　Windows上的命令提示符

(4) 尝试连接到MySQL客户端。

要进行连接，请输入第(2)步中的路径名再加上`-u` *`username`* `-p`。因此，命令行应该是

`c:\mysql\bin\mysql -u` *`username`* `-p` (Windows)

或

`/usr/local/mysql/bin/mysql -u` *`username`* `-p` (Unix 和 Mac OS X)

或

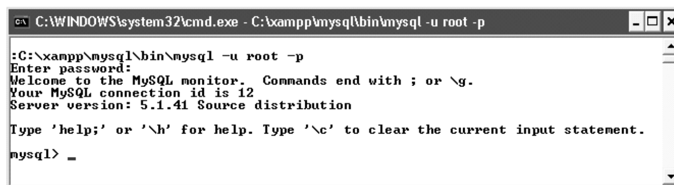
`C:\xampp\mysql\bin\mysql -u` *`username`* `-p` (Windows)

或

`/Applications/MAMP/Library/bin/mysql -u` *`username`* `-p` (Mac OS X)

将*`username`*替换为要使用的用户名。如果还没创建过其他用户，则这里可以用root（root是MySQL超级用户）。如果还没有为root用户设置密码，则可以忽略-p标志。

(5) 在提示符中输入密码（参见图A-16）。



图A-16　在Windows上成功地访问了MySQL客户端

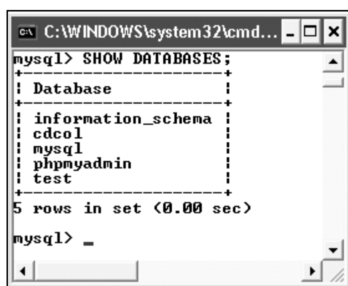
这里所必需的密码是再连接时针对用户名设置的MySQL密码。只有在第(4)步中使用了-p选项才会看到这个提示信息。

如果在Mac OS X上安装了MAMP，则root用户的密码就是root。如果在Windows上安装XAMPP，就没有初始密码。

(6) 列出可用的数据库（参见图A-17）：

```
SHOW DATABASES;
```

SHOW DATABASES命令是一个SQL查询，列出了寄宿在所安装的MySQL中并能由当前连接的用户看到的所有数据库。



图A-17　在全新安装MySQL之后，只能看到3个默认数据库

(7) 退出MySQL客户端。

键入exit或quit。

✓提示

- 如果看到Can’t connect to local MySQL server through socket…错误消息，通常表示MySQL没有运行。
- MySQL客户端是调试MySQL和PHP脚本的最佳工具。可以使用MySQL客户端来检查用户权限，并在PHP脚本之外运行查询。

A.4.2　使用phpMyAdmin

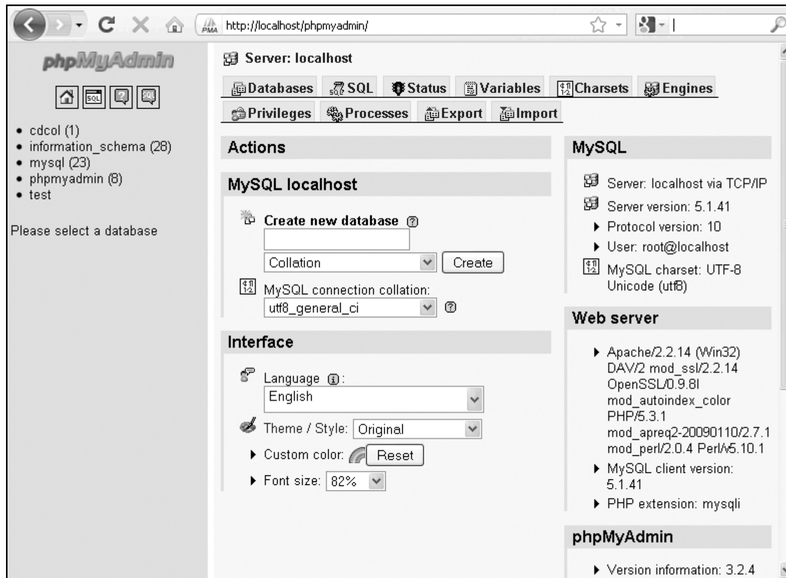
phpMyAdmin（www.phpmyadmin.net）是一个与MySQL交互的Web界面，它具有创建表、导入或导出记录等功能。phpMyAdmin是一个图形界面，它可能是用PHP编写的最流行的Web软件，几乎所有的PHP托管公司都提供这个软件。实际上，多合一安装包XAMPP和MAMP也包含这个软件。phpMyAdmin设计合理并且易于使用，下面我会介绍一些使用技巧。

⇛使用phpMyAdmin

(1) 在Web浏览器中访问phpMyAdmin（参见图A-18）。

如果用的是XAMPP，可以进入http://localhost/phpmyadmin/访问phpMyAdmin，也可以在控制面板中单击管理链接。如果用的是MAMP，可以进入http://localhost:8888/MAMP/访问phpMyAdmin。

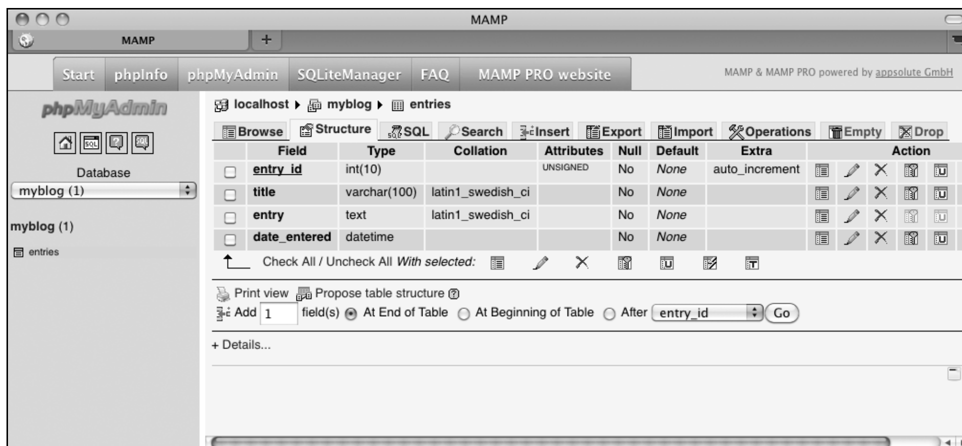
(2) 单击左列的数据库名，选择数据库。



图A-18　phpMyAdmin的主页

(3) 在左列中单击一个表名，选择该表（参见图A-19）。

这一步不是必需的，但是做这一步可以简化很多操作。



图A-19　在左列中选择数据库或表，在页面右边可以修改选项

(4) 使用页面右边的标签和链接可以执行一些常见任务。

一般而言，标签和链接是一些常用SQL命令的快捷方式。例如，`Browse`标签执行`SELECT`查询，而`Insert`标签会创建一个表单，用于添加记录。

(5) 使用SQL标签执行任意SQL命令。

你也可以使用SQL Query Window，它的链接在数据库和表名列表的上面。无论使用哪种界面，你都可以测试PHP中使用的查询语句，这样就无需增加脚本本身的复杂度了。

✓提示

- ❑ 还有很多与MySQL交互的客户端软件，但是MySQL命令行客户端和phpMyAdmin是最常用的两个软件。

A.5 管理 MySQL 用户

成功安装了MySQL之后，你就可以创建用户了。MySQL用户是一个基本的安全概念，可以限制访问、影响数据存储。需要说明的是，数据库应该有多个用户，就像操作系统一样。MySQL用户不同于操作系统用户。在你自己的计算机上学习PHP和MySQL时，你不需要创建新的用户，但是你需要在真正的网站服务器上创建MySQL用户，并赋予合适的权限。

最初安装的MySQL只有一个用户（名为root）并且没有设置密码（除非使用MAMP，它会设置一个默认密码root）。在安装完成后，你应该为该用户创建一个新密码。

在这之后，应该创建其他具备有限权限的用户。作为规则，不应该使用root用户来完成常规的日常操作。

A.5.1 为root用户设置密码

刚安装好MySQL时，没有建立任何有价值或安全的密码。这显然是一个安全风险，并且应该在使用服务器之前修正（因为root用户具有最高权限）。

可以使用phpMyAdmin或MySQL客户端修改用户的密码，但前提是必须运行MySQL服务器。如果MySQL当前没有运行，请使用本附录之前所介绍的步骤启动它。

其次，如果要修改root用户的密码，必须以root身份登录数据库。

⇛ 使用MySQL客户端为root用户设置密码

(1) 连接到MySQL客户端。

请参见之前介绍过的详细步骤。

(2) 输入下面的命令，将*thepassword*替换为想使用的密码（参见图A-20）：

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('thepassword ');
```

记住，MySQL的密码是区分大小写的，所以Kazan和kazan是不同的。在实际引用密码之前的PASSWORD术语告诉MySQL要对这个字符串进行加密。并且在`PASSWORD`和左括号之间不能有任何空白字符。

(3) 退出MySQL：

```
exit
```



图A-20　安装完软件之后，应该立即为root用户建立安全的密码

(4) 再次登录MySQL客户端，测试新密码。

既然密码已经建立起来了，就需要向连接命令中添加-p标志。将会看到Enter password:提示，在这里输入刚刚创建的密码即可。

⇛ 使用phpMyAdmin为root用户设置密码

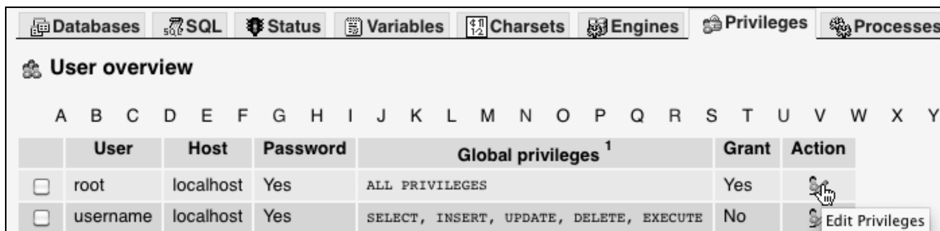
(1) 在Web浏览器中打开phpMyAdmin。

参见前面介绍的步骤了解详细信息。

(2) 在主页中，单击Privileges标签。

任何时候你都可以单击左上角的主页图标回到主页。

(3) 在用户列表中，在root用户行中单击Edit Privileges图标（参见图A-21）。



图A-21　phpMyAdmin显示的MySQL用户列表

(4) 使用结果页下面的Change Password 表单（参见图A-22）修改密码。

图A-22　phpMyAdmin中更新MySQL用户密码的表单

(5) 如果需要，在phpMyAdmin的配置文件中修改root用户的密码。

修改root用户的密码后，phpMyAdmin一般会阻止访问MySQL服务器。原因是在本地安装的phpMyAdmin一般会使用root用户连接MySQL，而root用户的密码被硬编码在了配置文件中。在完成步骤(1)~步骤(4)后，在phpMyAdmin目录[比如，/Applications/MAMP/bin/phpMyAdmin

（Mac OS X下的MAMP）或`C:\xampp\phpMyAdmin`（Windows下的XAMPP）]下找到`config.inc.php`文件。在任意文本编辑器或IDE中打开该文件，在以下代码中，用新密码替换旧密码：

```
$cfg['Servers'][$i]['password'] = 'the_new_password';
```

保存文件，在Web浏览器中重新加载phpMyAdmin。

A.5.2　创建用户和权限

在成功地安装和运行MySQL，并为root用户建立了密码之后，就该添加其他用户了。为了改善数据库的安全性，应该坚持创建新用户来访问数据库，而不是一直使用root用户。

MySQL权限系统用于确保在特定的数据库上授权特定的命令。例如Web主机可以利用其确保访问多个数据库的多个用户不会互相干扰。MySQL系统中的每个用户都可以访问特定主机（计算机）上特定数据库中的特定功能。root用户（MySQL的root用户，而不是操作系统的）具有最高的权限，可以用于创建子用户，然而子用户也可以具备像root用户那样的权限（这是很不可取的）。

当一个用户尝试在`mysql`服务器上做些什么时，MySQL首先会检查该用户是否有权限连接到服务器（基于用户名、用户密码和MySQL数据库中`user`表中的信息）。然后，MySQL会检查用户是否有权限在给定的数据库上运行特定的SQL语句——例如，选择数据、插入数据或创建新表。表A-1列出了可以为每个用户设置的权限。

表A-1　MySQL权限

权　　限	允许的操作
`SELECT`	从表中读取行
`INSERT`	向表中添加新数据行
`UPDATE`	修改表中现有数据
`DELETE`	从表中删除现有数据
`INDEX`	在表中创建和删除索引
`ALTER`	修改表的结构
`CREATE`	新建表或数据库
`DROP`	删除现有表或数据库
`RELOAD`	重新加载grant表（由此来发布用户变化）
`SHUTDOWN`	停止MySQL服务器
`PROCESS`	查看和停止现有的MySQL进程
`FILE`	从文本文件向表中导入数据
`GRANT`	新建用户
`REVOKE`	移除用户权限

在MySQL中设置用户和权限有很多种方式，但是我们建议使用MySQL客户端和`GRANT`命令手动完成。其语法类似于：

```
GRANT privileges ON database.* TO'username'@'hostname' IDENTIFIED BY 'password';
```

在该语句的*`privileges`*部分，可以列出来自表A-1中的特定权限，或者可以通过使用`ALL`

来指定所有的权限（这是很不明智的）。该语句的*database*.*部分指出了用户可以使用哪个数据库或表。可以使用*database*.*tablename*这样的语法来指定表的名字，或者使用*.*来指定所有的数据库（同样也是很不明智的）。最后，可以指定用户名和一个密码。

用户名的最大长度是16个字符。在创建用户名时，请确保避免使用空格（可以用下划线代替），并注意用户名是区分大小写的。

主机名是允许用户连接的计算机。它可以是域名，比如www.example.com或IP地址。一般来讲，localhost是限定的特定主机名，也就是说使用运行有MySQL数据库的计算机上的MySQL用户连接。要指定任意主机，可以在主机名中使用通配符（%）：

```
GRANT privileges ON database.* TO 'username'@ '%' IDENTIFIED BY'password';
```

我不推荐这种方式。在创建用户时，最好明确并限制权限。

密码没有长度限制，但也是区分大小写的。在MySQL数据库中密码是经过加密的，这意味着无法找回纯文本格式的密码。省略IDENTIFIED BY '*password*'子句可以不要求用户输入密码（但这也是应该避免的）。

在这个过程的示例中，我们将创建两个新用户，他们具有在temp数据库上的特定权限。记住，只能为用户设置在现有数据库上的权限。接下来的步骤还将介绍如何创建数据库。

在phpMyAdmin中创建用户

要在phpMyAdmin中创建用户，在phpMyAdmin主页中单击Privileges标签。在Privileges标签中，单击Add A New User表单定义用户名、主机、密码和权限。接下来单击Go。这样会创建一个具有普通权限的用户，但没有指定特定数据库的权限。

在结果页，选择数据库以便为用户分配权限，之后单击Go。在下一页，为用户分配对该数据库的特定权限，再单击Go。这就完成了给用户分配数据库权限的工作。这个程序可以很容易地为用户分配不同数据库的不同权限。

最后，返回主页中的Privileges标签，单击重载权限链接，使权限生效。

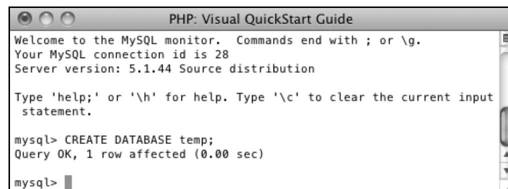
⇒ 使用GRANT创建新用户

(1) 作为root用户登录MySQL客户端。

这一步已经介绍过了。必须使用能够创建数据库和其他用户的用户登录。

(2) 创建一个新数据库（参见图A-23）：

```
CREATE DATABASE temp;
```

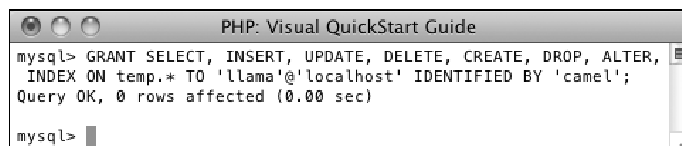


图A-23 创建新数据库

如果所用的MySQL服务器上还没有temp数据库，那么就使用CREATE DATABASE temp创建一个（后跟一个分号，这是MySQL客户端所必需的）。

(3) 在temp数据库中创建一个具有管理员级别权限的用户（参见图A-24）：

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, INDEX
→ON temp.* TO 'llama'@ 'localhost' IDENTIFIED BY 'camel';
```

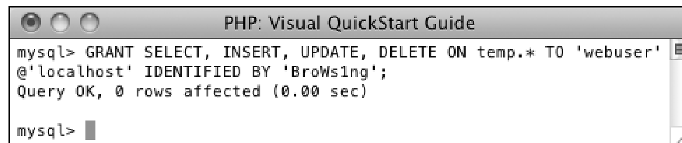


图A-24 为单个数据库创建管理员级别的用户

用户llama可以在temp数据库中进行创建表、修改表、插入数据、更新数据等操作。这基本上包括了除创建新用户之外的所有管理员级别的权限。请确保设置了密码——也许应该使用一个比这里更复杂的密码——另外，最好指定一个特定的主机。

(4) 创建一个对数据库只有基本访问权限的用户（参见图A-25）：

```
GRANT SELECT, INSERT, UPDATE, DELETE ON temp.* TO 'webuser'@ 'localhost'
→IDENTIFIED BY 'BroWs1ng';
```

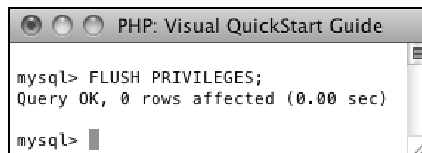


图A-25 对于同一个数据库，该用户具备更受限制的权限

现在，普通的webuser用户可以浏览记录（在表中SELECT），以及添加、编辑和删除记录，但该用户不能修改数据库的结构。当建立起用户和权限之后，应该采用自底向上的工作方式，在任何时候都启用尽可能少的访问权限。

(5) 应用更改（参见图A-26）：

```
FLUSH PRIVILEGES;
```



图A-26 在尝试用新创建的用户访问MySQL之前，请不要忘了这一步

在告诉MySQL重置其可接受的用户和权限列表之前，刚刚做出的修改不会产生影响，这也是这个命令要做的事情。忘记这一步会导致新创建的用户无法访问数据库，这是一种常见的错误。

✓提示

- 任何名字以test_开头的数据库都能够被有权限连接到MySQL的用户修改。因此，要小心不要用这种方式为数据库命名，除非它真的是用于实验的。
- REVOKE命令用于移除用户和权限。

附录 B

深入学习PHP的资源

本书面向初级PHP程序员，为他们的学习提供了一个良好的基础。由于更加专注于基础，因此本书忽略或跳过了一些主题。本附录列出了大量有用的与PHP相关的Internet资源，简要讨论了如何获取有关数据库和正文未讲述的主题的更多信息，并且给出了一些或新或旧的表格。

除了这里介绍的站点外，还应该知道本书配套网站的网址——www.LarryUllman.com。在这里可以找到本书用到的所有脚本，支持论坛和勘误页面等信息。

在编写本书第一版时，只有很少的高质量PHP站点，而现在则有差不多十几个（还有数以百计不那么好的）PHP站点。最好的、最显著的站点都已经列在这里了，但最好的资源永远是互联网上的搜索引擎。

B.1 PHP 在线资源

如果有特定于PHP的问题，应该能够很轻松地找到答案。本附录的这一节主要介绍可供使用的Internet上最好的资源。

B.1.1 PHP手册

所有的PHP程序员都应该在开始使用这门语言之前知道并且已经获取了某一版本的PHP手册。从PHP官方网站（www.php.net/docs.php）以及很多其他地方都可以得到该手册。

可以下载到各种语言、各种格式的手册。官方网站还提供了该手册的注释版www.php.net/manual/en/（英文），用户可以在这里添加有用的注解和评论。如果对某个特定的函数不是很了解，阅读手册并找到该函数的页面可能会得到答案。

在第1章我介绍过，通过www.php.net/*functionname*可以快速访问为任何特定函数提供的文档页面。例如，为`number_format()`函数提供的页面是www.php.net/number_format。

B.1.2 一般PHP网站

本节提到了在编程时可以访问的众多网站中的一部分，但需要自己探索才能发现最喜欢的网站。其中的大部分网站中都在其他PHP相关站点中有链接。很明显，首先要收藏的是PHP.net（www.php.net），这是PHP的官方站点。

应该熟悉Zend（www.zend.com），这是PHP核心创建者的主页。该站点包含了很多可下载的资源和其他大量的资源——可以说是来自专家们的第一手资料。

要查找特定主题的信息，PHPBuilder（www.phpbuilder.com）是一个不错的站点。该站点有大量关于如何使用PHP来完成特定任务的文章。此外，PHPBuilder还提供了支持论坛和代码库，开发者可以在这里上传示例脚本。

在PHP Zone（http://php.dzone.com）和php|architect（www.phparch.com）也可以找到很有用的文章，这些文章也经常会刊登在PHP的月刊杂志中。

W3Schools（www.w3schools.com）是一个很好的通用Web开发站点，但其中有大部分资源是面向PHP的。从使用PHP、HTML、CSS和JavaScript开发动态Web站点的角度看，这是一个非常理想的去处。

如果我发现了好的资源，不论是PHP资源或其他东西，我一般会将它们引用在自己的网站中。你可以在我的网站的PHP目录下找到它们，地址是：www.larryullman.com/category/php/。

B.1.3　代码仓库

如今在线的代码库只多不少。由于很多PHP程序员都是非常慷慨的（而且通常乐于分享），因此很多站点都整理了大量可供下载的PHP脚本。下面所示的是最好的在线代码仓库：

- HotScripts（www.hotscripts.com/PHP/）。
- PHP Resource Index（http://php.resourceindex.com）。
- PHP Classes Repository（www.phpclasses.org）。
- PX: the PHP Code Exchange（http://px.sklar.com）。

浏览Zend和PHPBuilder或搜索Web页面，也能找到代码示例。甚至还有一个专门用于查找代码的搜索引擎：www.koders.com。

B.1.4　新闻组和邮件列表

如果访问过新闻组，可以将其作为强大的宣传媒介，同时它还可以找到一些高难度问题的解决方法。当然，也可以向这些新闻组发表自己的专业见解，帮助那些有这方面需要的人。

最大的英语PHP新闻组是comp.lang.php。可以通过ISP或付费的Usenet组织来访问它。新闻组还提供除英语之外的其他语言支持。

PHP网站列出了可供注册的邮件列表，位于www.php.net/mailing-lists.php。

在向新闻组或邮件列表发帖之前，应该阅读一下Eric Steven Raymond所写的“How to Ask Questions the Smart Way”，参见www.catb.org/~esr/faqs/ smart-questions.html。花十分钟阅读该文档，便可以在寻求帮助时为你节省数小时的时间。

B.2　数据库资源

哪些数据库资源最有用，这明显取决于使用哪种数据库管理系统（DBMS）。最常与PHP一同

使用的数据库可能就是MySQL了，但PHP能够支持所有的标准数据库应用程序。

要学习如何使用MySQL，可以从MySQL的官方网站（www.mysql.com）开始。可以下载一份MySQL手册，作为工作时的参考资料。有些图书也是专门介绍MySQL的，包括我自己编著的*MySQL: Visual QuickStart Guide, 2nd Edition*（Peachpit Press, 2006）。

如果是用MySQL，别忘了下载并安装phpMyAdmin（www.phpmyadmin.net）。这是一款用PHP编写的，用于操作数据库的优秀工具。如果使用的是PostgreSQL或者甚至是Oracle，也可以找到类似的工具作为接口。每种数据库应用程序都有它自己的邮件列表和新闻组。

在数据库领域需要钻研的另一块资源就是SQL。下面列出了讨论SQL的网站，这种语言会用在各种数据库应用程序中：

- SQL Course（www.sqlcourse.com）
- A Gentle Introduction to SQL（www.sqlzoo.net）
- W3Schools’SQL Tutorial（www.w3schools.com/sql）
- SQL.org（www.sql.org）

我编著的《PHP6与MySQL5基础教程》一书也讨论了SQL和MySQL，并且比本书讲述得更详细。

B.3　Top 10 常见问题解答

调试是一种很有用的技能，这需要花时间并具备足够的经验才能掌握。为了不致于将读者带上一个装备不良的征程，本书将列出PHP脚本中最常见的10种问题，以及导致这些问题的最可能的原因。不过，首先本书会给出调试问题时的5个最佳建议。

(1) 知道正在使用的PHP的版本。

有些问题是特定于PHP的某个版本的。

在第一次使用一个服务器时，请使用`phpinfo()`脚本测试所使用的版本。另外还要确认所使用的MySQL的版本，如果可以的话，还应知道操作系统、Web服务器（如Apache 2.2）等。

(2) 通过URL运行所有的PHP脚本。

如果没有通过URL运行PHP脚本（而其中包含了向PHP脚本提交表单），则Web服务器不会处理请求，这意味着PHP不会执行任何代码。

(3) 相信错误消息！

很多初学者在解决问题时感到非常困难，因为他们不相信所看到的错误消息。尽管有些PHP错误消息很隐晦甚至容易让人误解，但如果PHP说第22行有问题，那么问题很可能就是在第22行。

(4) 避免通过“尝试”来解决问题！

如果不能确定是什么原因导致了问题，也不知道确切的解决方法是什么，请避免通过随机的尝试来解决问题。这样很可能会导致新问题的产生，并且只会进一步搞乱原有的问题。

(5) 休息一下！

本书能提供的最佳建议就是离开计算机休息一会儿，我通过这种方式解决了大量的问题。很

多时候真正需要的是一个清晰的大脑。

我们来继续，下面是在PHP中最可能遇到的10个问题。

(1) 空白页。

如果在提交了一个表单或加载了一个PHP脚本之后，在Web浏览器看到的是空白页，则很可能是由于某个错误中断了页面的执行。首先检查HTML源代码，看看是否有HTML错误。然后打开`php.ini`配置文件或PHP脚本中的`display_errors`，看看出现了什么PHP错误。

(2) undefined variable或undefined index错误（参见图B-1）。

Notice: Undefined index: Name in **/Users/larryullman/Sites/phpvqs4/handle_form.php** on line **16**

Thank you, Mr. , for your comments.

图B-1　对由拼写或大小写错误引发的未定义变量或索引错误的解释

只有当错误报告设置为最高级别时才会出现这些错误，它们可能指出了错误，也可能根本不是错误。请检查每个变量或数组索引的拼写，确保它们是正确的。然后，或者修改错误报告设置，或者在引用变量之前对其初始化。当然，还要确保变量的确带有值。

(3) 变量不带有值。

可能是在引用变量时写错了名字。请再次确认变量名的大小写和拼写是否正确，然后确保正确使用了`$_GET`、`$_POST`、`$_COOKIE`和`$_SESSION`。如果需要的话，使用`print_r()`函数来检查每个变量的名字和值。

(4) Call to undefined function…错误。

这种错误消息意味着试图使用一个在PHP中并不存在的函数。导致该错误产生的原因可能是函数名拼写错误、在调用函数前没有定义它或使用了所用PHP版本所不支持的函数。请检查拼写情况，并查看PHP手册中关于非用户定义函数的说明来查找问题所在。

(5) Headers already sent错误（参见图B-2）。

该错误消息表明在Web浏览器已经接收到HTML或一个空白页之后调用了HTTP头相关的函数——`header()`、`setcookie()`或`session_ start()`。请复查在调用这些函数之前脚本中发生了什么，或使用输出缓冲来避免混乱。还可以利用输出缓冲来避免这些错误的出现。

Warning: Cannot modify header information - headers already sent by (output started at /Users/larryullman/Sites/phpvqs4/templates/header.html:4) in **/Users/larryullman/Sites/phpvqs4/login.php** on line **22**

图B-2　某些函数因为调用时机错误而导致的headers already sent错误

(6) Access denied错误（参见图B-3）。

如果在尝试使用数据库时看到了这个错误消息，则你正在使用的用户名、密码和主机这个三元组可能不具备访问数据库的权限。这通常不是PHP的问题。请确认正在使用的值，并尝试用一个不同的系统（如MySQL客户端）去连接数据库。



图B-3　如果MySQL访问信息不正确，则会看到数据库访问被拒绝的消息

(7) Supplied argument is not a valid MySQL result resource错误。

这是另一个数据库相关的错误消息，该消息意味着不恰当地使用了一个查询结果，这通常是由于试图在一个没有返回任何记录的查询中检索行。要解决这一问题，请找到运行的查询，并使用其他工具（如MySQL客户端或phpMyAdmin）测试。另外还请检查变量名是否正确。

(8) 预置的HTML表单值被截断。

必须将HTML表单中文本框控件的`value`属性值放置在双引号中。如果不这么做，只有第一个空格之前的部分会被设置为文本框控件的值。

(9) 条件或循环语句产生非预期行为。

这些逻辑错误非常常见。请确定没有用错运算符（如错将==写成=），以及引用的变量确实正确。然后使用`print`语句检查脚本做了些什么。

(10) 解析错误（参见图B-4）。

解析错误将是要处理的最常见的错误。即便是PHP编程老手，也会偶尔遇到这个问题。请检查每条语句是否都以分号结尾，以及所有的引号、圆括号、方括号和花括号都是配对的。如果还是没能找到解析错误，可以使用`/*`和`*/`字符注释掉大段代码。然后每次取消一小部分的注释，直到再次看到解析错误。然后就能知道脚本中哪段代码出问题了（或最可能的地方）。

Parse error: syntax error, unexpected T_ENCAPSED_AND_WHITESPACE, expecting T_STRING or T_VARIABLE or T_NUM_STRING in **/Users/larryullman/Sites /phpvqs4/handle_form.php** on line **19**

图B-4　解析错误非常常见，而且能阻止脚本的执行

B.4　下一步

本书只是帮助你开始使用PHP，但在这之后，还有一些主题需要研究。

B.4.1　安全

Web服务器、操作系统、数据库和PHP的安全这些主题都值得拥有各自的书籍。尽管本书尝试介绍了安全Web应用程序的编写，但这个领域依然有很多区域是需要自己去探索的。可以通过

下面这些站点开始你的探索：

- ❑ Scarlet学习（www.securereality.com.au/studyinscarlet.txt）；
- ❑ W3C安全资源（www.w3.org/Security）。

第一个站点是关于一篇编写有关安全PHP代码的文章。第二个站点是World Wide Web Consortium针对Web相关的安全性问题提供的资源页。

还需要阅读PHP手册和所使用的数据库手册中的相关章节。在Internet上搜索PHP和security也会找到很多有趣的文章。

B.4.2　面向对象编程

本书没有涵盖对象和面向对象编程（OOP）的主题，这出于以下两个原因：

(1) 它还是远远超过了初学者指南的范畴；

(2) 不理解对象，并不会限制理解PHP能够做什么。

如果决定要学习这一主题，可以在PHP站点中搜索一些教程，找一个框架（参见本附录的B.4.3节），或阅读一下我编著的*PHP 5 Advanced: Visual QuickPro Guide*（Peachpit Press，2007）。我在其中用了大约150页的篇幅专门介绍OOP（但仍然有大部分的OOP主题没有涉及）！

B.4.3　框架

框架是一种已经建立好的代码库，用于开发成熟的Web应用程序。通过重用已经由其他人证实过的PHP代码，可以快速地建立一部分或全部Web站点。

有很多PHP框架可供使用，可以从PEAR开始。PEAR是一个巨大的PHP代码仓库，它是使用对象（从技术上说是类）编写的。即便自己不使用对象，或是刚刚理解这一概念，依然能从PEAR网站（http:// pear.php.net）获得大量有价值的信息。PEAR及其Web站点提供了免费的、漂亮的代码并展示了良好的PHP编码风格。PECL（http://pecl. php.net）是PEAR的更为强大的姊妹库。

另一个需要考虑的框架是Zend Framework（http://framework.zend.com）。这个框架相对较新，具有大量优点，并且有编写完善的文档。

在写作本书时，我个人非常喜欢的PHP框架是Yii（www.yiiframework.com）。在我的网站中我写了很多有关Yii的文章。

很多人喜爱框架和它们提供的功能。但另一方面，学习使用一个框架确实要花一些时间，而且自定义框架的行为可能是一件让人畏惧的事情。

B.4.4　JavaScript和Ajax

JavaScript是运行在Web浏览器中的客户端技术。它可以用于向Web站点增添大量动态特性，从简单的诸如图片翻滚的视觉效果，到交互式的菜单和表单。由于JavaScript是运行在Web浏览器中的，因此它能提供一些PHP所不能提供的功能。不过，JavaScript和PHP一样，也是相当易学易用的。更多信息，请参见：

- JavaScript.com（www.javascript.com）；
- W3School上的JavaScript页（www.w3schools.com/js/）。

Ajax（可能意味着异步的JavaScript和XML，也可能不是，取决于问的是谁）自2005年以来已经成了Web开发社区里最热门的话题。这项技术使用JavaScript与服务器进行通信，但用户并不能察觉到。最终的效果是，Web站点的行为越来越像桌面应用程序。

更多信息，请参见：Ajaxian（www.ajaxian.com）；或在网络上搜索相关信息。

我强烈建议你学习一下jQuery（www.jquery.com），它可以满足你对JavaScript、Ajax和其他动态Web的需求。jQuery是一个JavaScript框架，非常容易使用，功能也非常强大，而且有详细的说明文档。

B.4.5　其他书籍

我希望读者在读完本书之后，能对学习更多的PHP和一般Web开发知识感兴趣。尽管我可以推荐其他作者的书，但这里有个固有的冲突，而且我认为对于相同的读者群来说那些作者是竞争对手。因此，我只着重介绍了我编著的其他几本书，以及这些书与本书的对比情况。

《PHP6和MySQL5基础教程》是本书的后续（该书的最新版是命名为*PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide，Fourth Edition*）。两本书有些内容重叠，尤其是在前几章，但它与本书使用的例子是不同的，而且比本书的讲述步伐更快。特别是涵盖了更多有关MySQL和SQL方面的内容，并且有三个不同的示例章节：一个多语言论坛、一个用户注册和登录系统以及一个电子商务站点。

刚刚提到过，我编著的*PHP 5 Advanced: Visual QuickPro Guide*（Peachpit Press, 2007）也是本书的一个续集。这本书的内容更为高级，用大量的篇幅讲述了诸如OOP和PEAR这样的主题。这本书并不需要顺序阅读，这与本书不同，它的每一章都着重于一个特定的主题。

MySQL: Visual QuickStart Guide, Second Edition（Peachpit Press, 2006）这本书看上去是专门讲解MySQL和SQL的。尽管其中有4章涵盖了与MySQL交互的语言——PHP、Perl和JavaScript，外加一个技术章节，但这本书还是主要用来介绍MySQL的安装、管理以及MySQL知识。

我的新书（在写这本书之前完成的）是*Effortless E-Commerce with PHP and MySQL*（New Riders，2011）。这本书介绍了创建功能完备的电子商务网站所需要的全部知识。书中使用了两个具体的示例并且将两个不同的支付系统合并在一起。当然，也是用PHP和MySQL实现的。

最后，我编著的*Building a Web Site with Ajax: Visual QuickProject Guide*（Peachpit Press, 2008）一书介绍了编写一个Ajax风格的Web站点的全过程。这本书也使用了PHP和MySQL，但没有按照讲解JavaScript和Ajax的方式去讲解这些技术。

B.5　表

本书中散布着不少表格，为了便于参考，这里重新列出了其中最重要的3个表格。此外，还加了一个新的表格，列出了运算符优先级（表B-1）。这部分运算符是按照优先级从高到低列出的，

例如，乘法的优先级高于加法。

表B-1　运算符优先级

<table>
<tr><th>运算符优先级</th><th>运算符优先级</th></tr>
<tr><td>! ++ --</td><td>||</td></tr>
<tr><td>* / %</td><td>= += -= *= /= .= %=</td></tr>
<tr><td>+ - .</td><td>and</td></tr>
<tr><td>< <= > >=</td><td>xor</td></tr>
<tr><td>== != ===</td><td>or</td></tr>
<tr><td>&&</td><td></td></tr>
</table>

表B-2列出了PHP的主要运算符和它们的类型。最重要的是要记得单个等号（=）用于为变量赋值，而双等号（==）用于判断相等性。

表B-2　PHP运算符和使用类型

<table>
<tr><th>运　算　符</th><th>用　　法</th><th>类　　型</th></tr>
<tr><td>+</td><td>加</td><td>算术</td></tr>
<tr><td>-</td><td>减</td><td>算术</td></tr>
<tr><td>*</td><td>乘</td><td>算术</td></tr>
<tr><td>/</td><td>除</td><td>算术</td></tr>
<tr><td>%</td><td>模（除法的余数）</td><td>算术</td></tr>
<tr><td>++</td><td>增量</td><td>算术</td></tr>
<tr><td>--</td><td>减量</td><td>算术</td></tr>
<tr><td>=</td><td>将值赋给变量</td><td>赋值</td></tr>
<tr><td>==</td><td>相等</td><td>比较</td></tr>
<tr><td>!=</td><td>不相等</td><td>比较</td></tr>
<tr><td><</td><td>小于</td><td>比较</td></tr>
<tr><td>></td><td>大于</td><td>比较</td></tr>
<tr><td><=</td><td>小于或等于</td><td>比较</td></tr>
<tr><td>>=</td><td>大于或等于</td><td>比较</td></tr>
<tr><td>!</td><td>取反</td><td>逻辑</td></tr>
<tr><td>AND</td><td>与</td><td>逻辑</td></tr>
<tr><td>&&</td><td>与</td><td>逻辑</td></tr>
<tr><td>OR</td><td>或</td><td>逻辑</td></tr>
<tr><td>||</td><td>或</td><td>逻辑</td></tr>
<tr><td>.</td><td>或非</td><td>逻辑</td></tr>
<tr><td>XOR</td><td>连接</td><td>字符串</td></tr>
<tr><td>.=</td><td></td><td></td></tr>
<tr><td>+=</td><td></td><td></td></tr>
<tr><td>-=</td><td></td><td></td></tr>
</table>

表B-3指出了在打开一个文件时可以使用的模式。你的选择决定了PHP可以在该文件上进行的操作——写入、读取等。

由于很难记住`date()`函数的多种格式，因此在使用`date()`函数时，请把表B-4放在旁边。

表B-3 `fopen()`的模式及其意义

模 式	含 义
`r`	只读；从文件的起始位置开始读取
`r+`	读或写；从文件的起始位置开始
`w`	只写；如果文件不存在则创建文件，并且盖写现有内容
`w+`	读或写；如果文件不存在则创建文件，并且盖写现有内容（写入时）
`a`	只写；如果文件不存在则创建文件，将新数据追加到文件末尾（保留任何现有数据并向其中添加新数据）
`a+`	读或写，如果文件不存在则创建文件，将新数据追加到文件末尾（写入时）
`x`	只写；如果文件不存在则创建文件，如果文件存在则什么也不做（并发出一个警告）
`x+`	读或写；如果文件不存在则创建文件，如果文件存在则什么也不做（并发出一个警告）（写入时）

表B-4 `date()`函数的格式化字符及其意义与示例

字 符	含 义	示 例
`Y`	4位数字表示的年份	2011
`y`	2位数字表示的年份	11
`L`	是否为闰年	1（表示“是”）
`n`	1或2位数字表示的月份	2
`m`	2位数字表示的月份	02
`F`	月份	February
`M`	3个字母表示的月份	Feb
`j`	1或2位数字表示的月份中的一天	8
`d`	2位数字表示的月份中的一天	08
`l`（小写的L）	一周中的某一天	Monday
`D`	3个字母表示的一周中的某一天	Mon
`w`	1位数字表示的一周中的某一天	0（星期日）
`z`	一年中的某一天：0~365	189
`t`	月份中有多少天	31
`S`	2个字符表示的天数英文序数词后缀	rd
`g`	小时数，1或2位数字表示的12小时制格式	6
`G`	小时数，1或2位数字表示的24小时制格式	18
`h`	小时数，2位数字表示的12小时制格式	06
`H`	小时数，2位数字表示的24小时制格式	18
`i`	分钟数	45
`s`	秒数	18
`u`	毫秒数	1234
`a`	am或pm	am
`A`	AM或PM	PM
`U`	从epoch开始的秒数	1048623008
`e`	时区	UTC
`I`（大写的i）	是否为夏令时	1（表示“是”）
`O`	与GMT之间的时差	+0600

TURING 图灵教育 站在巨人的肩上

读者积分赠书卡

手机号码：________________（此为会员编号）

姓　　名：________________　　性别：□男　□女　　出生年月：____年___月

通信地址：____

邮政编码：____

电子邮件：____

您购买的图书是：26093 /《PHP基础教程（第4版）》

（65.00元）

您获得的会员积分是：6.5 分

欢迎参加“**有奖DEBUG**”活动。提交本书勘误，每确认一处即可获赠积分5分。详情见图灵网站。

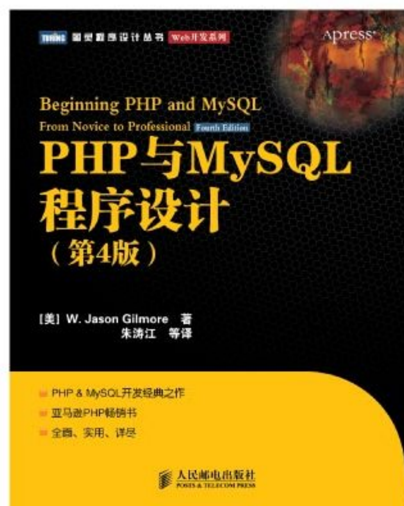
请沿虚线剪下此页，寄回图灵公司，即可成为图灵读者俱乐部的一员（复印无效）。**积分累计，可获赠书**（赠书清单见图灵网站）。

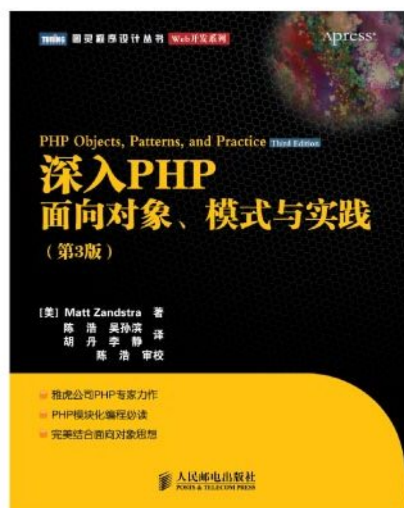
邮政编码：　100107

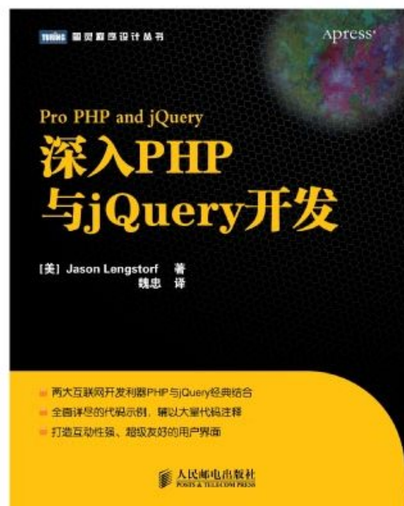
通信地址：　北京市朝阳区北苑路 13 号院 1 号楼 C603

北京图灵文化发展有限公司　图灵读者俱乐部




图灵程序设计丛书
Web开发系列
Apress
Beginning PHP and MySQL
From Novice to Professional Fourth Edition
PHP与MySQL
程序设计
（第4版）
[美] W. Jason Gilmore 著
朱涛江 等译
PHP & MySQL开发经典之作
亚马逊PHP畅销书
全面、实用、详尽
人民邮电出版社
POSTS & TELECOM PRESS


图灵程序设计丛书
Web开发系列
Apress
PHP Objects, Patterns, and Practice Third Edition
深入PHP
面向对象、模式与实践
（第3版）
[美] Matt Zandstra 著
陈 浩 吴孙滨
胡 丹 李 静 译
陈 浩 审校
雅虎公司PHP专家力作
PHP模块化编程必读
完美结合面向对象思想
人民邮电出版社
POSTS & TELECOM PRESS


图灵程序设计丛书
Apress
Pro PHP and jQuery
深入PHP
与jQuery开发
[美] Jason Lengstorf 著
魏忠 译
两大互联网开发利器PHP与jQuery经典结合
全面详尽的代码示例，辅以大量代码注释
打造互动性强、超级友好的用户界面
人民邮电出版社
POSTS & TELECOM PRESS

“Larry Ullman写的这本PHP书体现他两个重要的才能：（1）他能够让你对PHP有全面的了解；（2）他能够用简明的语言阐明技术内容。这又是一部优秀的著作！”

——资深计算机取证专家Jerry Saperstein

“我的PHP知识很有限，通读完这本书又做了很多书上的练习之后，我学会了。至少对我来说，这本书实现了其承诺，即快速、便捷地学会PHP。”

——硅谷高级管理人员和技术顾问Bruce B. Razban

PHP for the Web Visual QuickStart Guide Fourth Edition

PHP基础教程（第4版）

PHP语言简单易学、功能强大、成本低廉、开发高效、执行灵活，是开发Web应用程序的理想工具。

本书是知名的PHP入门图书，书中并不包含深奥的理论或者高级应用，而是以大量来自实战的例子、屏幕图和详细的解释，以通俗的语言介绍了PHP的各项基础知识。第4版最大的变化在第13章，教你一步步创建一个功能齐全的网站。

无论你是否拥有编程经验，阅读本书都会帮助你加深对PHP的理解，提高你的工作成效。让我们一起踏上轻松愉快的PHP之旅，建设迷人的网络世界吧。



图灵社区：www.ituring.com.cn
反馈/投稿/推荐信箱：contact@turingbook.com
热线：(010)51095186转604

分类建议 计算机/网络技术/PHP

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-26093-2
定价：65.00元